VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Řešení optimalizačních inženýrských úloh pomocí efektivních simulačních metod

Cross-entropy method as a tool for solution of optimization problems

2015

Bc. Michal Béreš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání diplomové práce

Student: **Bc. Michal Béreš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1103T031 Výpočetní matematika

Téma: Řešení optimalizačních inženýrských úloh pomocí efektivních simulačních metod
Cross-entropy method as a tool for solution of optimization problems

Zásady pro vypracování:

Reálné inženýrské optimalizační úlohy jsou zřídka řešitelné analyticky, většinou se řeší na bázi simulačních metod. Relativně nová simulační technika je založena na CE algoritmu (Cross-Entropy), což představuje inovovanou Monte Carlo techniku, která slouží pro řešení složitých kombinatorických optimalizačních problémů. Metoda má aplikace v řadě oblastí, jako je například teorie grafů, počítačová biologie, uspořádání DNA, atd.

Postup práce:
1. Formulace optimalizačního problému
2. Dostupné simulační algoritmy pro řešení optimalizačních úloh.
3. CE algoritmus a jeho modifikace pro řešení kombinatorické optimalizace, řešení asociovaného stochastického problému.
4. Další aplikace a modifikace CE algoritmu pro řešení inženýrských úloh, podle pokynů vedoucího DP.
5. PC implementace nově nalezených algoritmů.

Seznam doporučené odborné literatury:

R.Y. Rubinstein and D.P. Kroese. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning,
Springer-Verlag, New York, 2004.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Radim Briš, CSc.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych na tomto místě poděkoval vedoucímu mé diplomové práce prof. Ing. Radimu Brišovi, CSc. za čas, který mi věnoval při konzultacích, a také za zapůjčení potřebné literatury.

Abstrakt

Tato diplomová práce se zabývá využitím Cross-entropy metody pro efektivní řešení optimalizačních úloh a simulaci *rare events* (RE) systémů. Cílem práce je aplikace Cross-entropy metody na reálnou RE úlohu z odvětví finančních rizik, řešení spojitých optimalizačních problémů a kombinatorických optimalizačních problémů, konkrétně problém obchodního cestujícího, max-cut problém a problém zarovnání sekvencí. Důležitou součástí práce je paralelní implementace výše zmíněných úloh na GPU. Hlavními nástroji pro implementaci zmíněných problémů jsou Matlab 2014b a CUDA toolkit 6.0.

Klíčová slova: Cross-entropy metoda, simulační metody, optimalizační problémy, Monte Carlo metody, CREC model, problém obchodního cestujícího, max-cut problém, problém zarovnání sekvencí, GPU paralelizace

Abstract

This master thesis focuses on using Cross-entropy method as a tool for solving optimalization problems. The aim of the work is the application of Cross-entropy method on the Credit Risk Economic Capital model, continuous optimization problems and combinatorial optimization problems, specifically the traveling salesman problem, max-cut problem and the sequence alignment problem. An important part of this thesis is the parallel implementation of the aforementioned tasks on the GPU. The main tools for implementing all mentioned problems are Matlab 2014b and CUDA 6.0 toolkit.

Keywords: Cross-entropy method, simulation methods, optimization problems, Monte Carlo methods, CREC model, traveling salesman problem, max-cut problem, sequence alignment problem, GPU parallelization

Seznam použitých zkratek a symbolů

ASP	–	Asociovaný stochastický problém
CE	–	Cross-Entropy
cdf	–	kumulativní distribuční funkce
CMC	–	Crude Monte Carlo
CREC	–	Credit Risk Econimic Capital
CUDA	–	Compute Unified Device Architecture
EAD	–	Exposure At Default
FACE	–	plně adaptivní CE metoda (*"Fully Adaptive CE"*)
IS	–	Importance Sampling
K-L	–	Kullback-Leibler
LGD	–	Loss Given at Default
LR	–	Likelihood Ratio
MC	–	Monte Carlo
MŘ	–	Markovův řetězec
MCWR	–	MŘ s nahrazováním (*"Markov Chain With Replacement"*)
NEF	–	rodina přirozeně exp. rozdělení (*"Natural Exponential Family"*)
RE	–	Řídké jevy (*"Rare Events"*)
SEN	–	stochastická síť daná hranami (*"Stochastic Edge Network"*)
SNN	–	stochastická síť daná vrcholy (*"Stochastic Node Network"*)
SC	–	Stochastic Counterpart
TSP	–	problém obchodního cestujícího (*"Traveling Salesman Problem"*)
VM	–	minimalizace rozptylu (*"Variance Minimalization"*)
$\boldsymbol{X}$	–	náhodná veličina
W	–	Likelihood Ratio
$\mathcal{D}$	–	Kullback-Leiblerova divergence
$\mathcal{N}$	–	normální rozdělení pravděpodobnosti
$\mathcal{U}$	–	uniformní rozdělení pravděpodobnosti
$\mathbb{E}_f$	–	střední hodnota vzhledem hustotě pravděpodobnosti f
Θ	–	množina parametrů pravděpodobnostního rozdělení
$\mathbb{V}$	–	parametrická rodina pravděpodobnostních rozdělení
α	–	vyhlazovací parametr
$\mathcal{X}$	–	množina přípustných hodnot

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

S rozvojem moderní výpočetní techniky se naskýtá možnost řešit stále rozsáhlejší úlohy. Jednou z hlavních technik řešení rozsáhlých úloh jsou takzvané Monte Carlo metody. Tyto metody jsou založeny na generování náhodných čísel a simulačním přístupu k řešení úloh, úvod do Monte Carlo problematiky lze nalézt například v [17]. Hlavní výhodou Monte Carlo technik je fakt, že se jedná o *„embarrassingly parallel"*[1] problémy, což znamená možnost poměrně jednoduché a efektivní paralelizace. Jednou z nových a v poslední době velmi rozvíjených Monte Carlo technik je Cross-entropy metoda. Touto metodou v aplikaci na optimalizační úlohy a simulaci rozsáhlých systémů se budeme v této práci zabývat.

Cílem práce je seznámit se s Cross-entropy metodou a využít její modifikace pro řešení optimalizačních problémů, především takzvaných kombinatorických optimalizací. Dalším cílem této práce je využití Cross-entropy metody pro urychlení simulace CREC modelu, který slouží bankám pro kvantifikaci finanční zálohy za účelem solventnosti. V neposlední řadě je cílem práce implementovat veškeré probírané metody pomoci jazyku Matlab a vybrané modely paralelizovat pomocí GPU a technologie CUDA.

Práce je rozdělena na několik částí, které čtenáře postupně dovedou od formulace optimalizačního problému v sekci 2 až po efektivní řešení vybraných optimalizačních problémů Cross-entropy metodou v sekcích 6.1, 6.2, 6.3 a 7. Sekce 3 je věnována odvození Cross-entropy metody a základní teorii. Sekce 4 se pak věnuje aplikaci metody při simulaci systémů s řídkými jevy (*„rare events"*) a sekce 5 rozebírá postup aplikace Cross-entropy metody na optimalizační problémy.

Při studiu této práce je předpokládána znalost teorie pravděpodobnosti a statistiky, základů teorie grafů a teorie míry. Pro studium statistiky doporučujeme [14, 12, 13], teorie grafů [16] a teorie míry [15].

[1] *„embarrassingly parallel"* problémem rozumíme úlohu složenou z mnoha malých výpočetně nezávislých částí

2 Motivace k optimalizačním úlohám

Tato část čerpá především z [23, 2]. Optimalizační úlohy jsou jednou z nejčastějších aplikací matematiky. V téměř všech odvětvích se lze setkat s problémy, kdy chceme maximalizovat profit nebo minimalizovat náklady na výrobu, což vede k optimalizačním úlohám. Obecně lze tyto metody formulovat jako řešení úlohy

$$\max_{x\in\mathcal{X}} f(x), \text{ nebo } \min_{x\in\mathcal{X}} f(x),$$

kde $f(x)$ je cenová funkce, která na základě parametru x udává velikost profitu/nákladů, a $\mathcal{X}$ je množina přípustných stavů, která udává možnosti mezi kterými daný optimální parametr hledáme.

Princip optimalizačních úloh si ukážeme na krátkém motivačním příkladu.

Příklad 2.1. Farmář má prase vážící 100kg. Každý další den chovu stojí 45 Kč a zvíře za něj přibere 2kg. Cena, kterou při prodeji utrží, je 65Kč za kilogram, avšak za každý další týden cena klesá o 1Kč za kg. Za kolik dní od teď se vyplatí farmáři prase prodat?

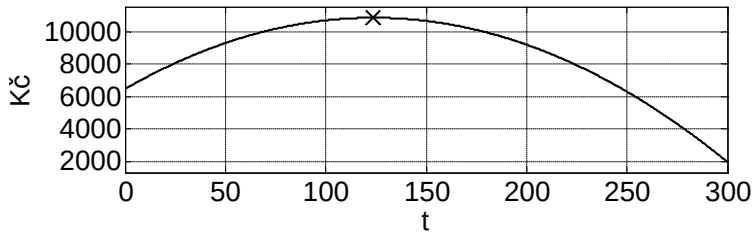
Nejprve formulujme matematickou úlohu. Cenovou funkcí bude zřejmě výsledný zisk, který se rovná částce utržené za prodej mínus nákladům na chov

$$f(t) = \left(65 - \frac{t}{7}\right) \cdot w - 45 \cdot t,$$

kde t značí čas ve dnech a $w = 100 + 2 \cdot t$ je váha zvířete. Úloha na optimalizaci je tedy následující

$$\max_{t\in(0,\infty)} \left(65 - \frac{t}{7}\right) \cdot (100 + 2 \cdot t) - 45 \cdot t.$$

Grafické znázornění vývoje profitu v závislosti na čase lze vidět na následujícím obrázku.



Obrázek 2.1: Vývoj profitu farmáře v závislosti na čase

Jelikož je cenová funkce $f(t)$ kvadratickou funkcí se záporným koeficientem u kvadratického členu, řešení optimalizačního problému nalezneme jako lokální extrém funkce $f(t)$. Řešení bude tedy

$$f(t)' = 0 \rightarrow \left(\left(65 - \frac{t}{7}\right) \cdot (100 + 2 \cdot t) - 45 \cdot t\right)' = 0$$

$$-\frac{4}{7}t - \frac{100}{7} + 130 - 45 = 0 \rightarrow t = 123 + \frac{3}{4}.$$

Optimální čas prodeje je tedy $123 + \frac{3}{4}$ dne, proto stačí ověřit hodnoty $f\,(123) = 10875.3$ a $f\,(124) = 10875.4$, čímž jsme získali řešení úlohy. Optimální čas pro prodej je tedy za 124 dní, přičemž profit bude 10875.4Kč. △

Tato úloha byla příkladem na spojitou optimalizaci. Simulačními technikami pro řešení rozsáhlých spojitých optimalizačních úloh se budeme zabývat v sekci 7.

Jiným typem optimalizačních úloh jsou úlohy na kombinatorické optimalizace. Příkladem takovéto úlohy je problém umístění 8 královen na šachovnici, tak aby se žádné dvě neohrožovaly, probíraný na konci sekce 5. V těchto optimalizačních úlohách jednotlivé veličiny nabývají pouze několika diskrétních stavů, avšak je jich mnoho. Těmto úlohám se budeme v této práci věnovat především.

Jako hlavní techniku pro řešení optimalizačních problémů budeme používat simulační Cross-entropy metodu, jejíž základy jsou odvozeny v následující kapitole.

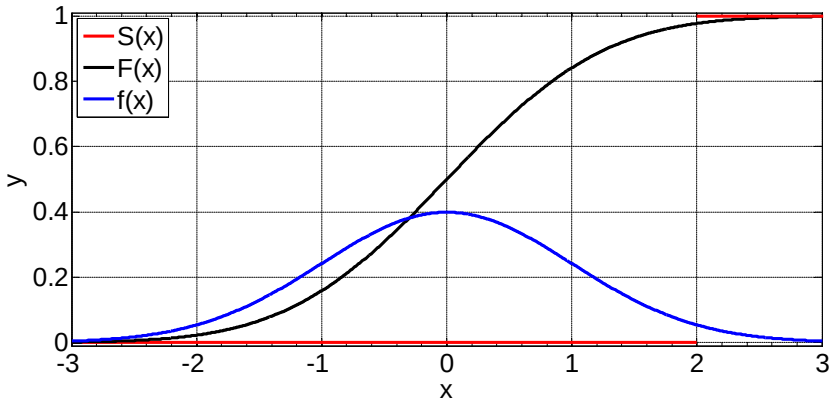
3 Cross-entropy metoda

Následující text čerpá z [1, 7, 4, 5, 3]. V mnoha aplikacích simulačních metod je k dosažení požadované přesnosti nutné použít značný počet pokusů. Tato obtíž se dá často vyřešit využitím tzv. metod redukce rozptylu. Obecně existuje mnoho přístupů, jak snížit rozptyl, tato práce je však zaměřena pouze na jeden a to *importance sampling* (IS). Princip IS a jeho souvislost s CE metodou bude vysvětlena dále v této kapitole.

Nejprve uveďme motivační úlohu, na které bude později demonstrována efektivita metody.

Příklad 3.1. Mějme rozdělení pravděpodobnosti dané jeho hustotou pravděpodobnosti $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^n$ a funkci $S(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. Úkolem je zjistit střední hodnotu $\mathbb{E}(S(\boldsymbol{X})), \boldsymbol{X} \sim f$.

Pro jednoduchost a možnost analytického řešení zvolme rozdělení pravděpodobnosti $\mathcal{N}(0,1)$ a zkoumanou funkci $S(x) = I(x > 2)$. Je nutné dodat, že v praxi může být funkce $S(\boldsymbol{x})$ velice složitá a její hodnota může být získána pouze simulačně jako u příkladu v kapitole 4.4.



Obrázek 3.1: Motivační úloha: Funkce $S(x)$, hustota pravděpodobnosti a cdf

Analytické řešeni je v tomto případě jednoduché, stačí spočítat hodnotu distribuční funkce $\mathcal{N}(0,1)$ v bodě 2 a odečíst od jedné: $1 - F(2) = 0.0228$. △

3.1 Importance sampling

Při použití simulačních metod je mnohdy třeba určit střední hodnotu $\mathbb{E}(S(\boldsymbol{X}))$. Často se však bude jednat o problém, kde hodnota $S(\boldsymbol{X})$ je velmi malá mimo nějakou množinu Ω přičemž $\mathbb{P}(\boldsymbol{X} \in \Omega)$ bude také malá. Toto může být způsobeno například malou velikostí množiny Ω nebo nízkou pravděpodobností, že stav v množině Ω nastane (častý příklad chvostu[2] rozdělení pravděpodobnosti). V případě klasické MC metody bude vzorků v Ω velmi málo (nebo dokonce žádný) a tedy i odhad výsledku bude špatný.

[2] anglicky *„tail of distribution"*

Je zřejmé, že k zajištění lepšího odhadu výsledku, je třeba zajistit větší počet pokusů z oblasti Ω. Tohoto lze dosáhnout pomocí generování vzorků z jiného rozdělení, zvoleného s ohledem na množinu Ω, což odpovídá definici IS metody.

Pomocí IS metody lze docílit značného zpřesnění odhadu a vyřešit problémy, které by byly obyčejnou metodou MC neřešitelné. Může se však stát, že při nevhodném využití IS dojde ke zhoršení odhadu. Problémům, které mohou nastat při použití IS se věnuje sekce 3.4.

Značení přebíráme z [1, str. 3].

Značení 3.2. Pravděpodobnostním rozdělením budeme dále rozumět míru ν definovanou jako

$$\nu(B) = \mathbb{P}(X \in B), B \in \mathcal{B},$$

kde $\mathcal{B}$ je systém Borelovských množin nad $\mathbb{R}^n$. Pokud se jedná o diskrétní rozdělení s pravděpodobnostní funkcí f, pak

$$\nu(B) = \sum_B f(\boldsymbol{x}),$$

a pokud se jedná o spojité rozdělení s hustotou pravděpodobnosti f, pak

$$\nu(B) = \int_B f(\boldsymbol{x}) \, \mathrm{d}\lambda.$$

V obou těchto případech je f hustotou ν vzhledem k nějaké základní míře μ a tedy v diskrétním i spojitém případě můžeme psát

$$\int_B f(\boldsymbol{x}) \, \mu(\mathrm{d}\boldsymbol{x}), B \in \mathcal{B}. \tag{3.1}$$

V diskrétním případě se bude jednat o aritmetickou míru a ve spojitém případě o Lebesgueovu míru. Dále v textu budeme integrál ze vzorce (3.1) značit pouze

$$\int_B f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}.$$

Více o mírách a Lebesgueovu integrálu lze nalézt v [15].

Je dána úloha na určení střední hodnoty funkce

$$\ell = \mathbb{E}_f(S(\boldsymbol{X})) = \int_{\mathcal{D}} S(\boldsymbol{x}) \cdot f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x},$$

kde $f(\boldsymbol{x})$ je hustota pravděpodobnosti definována na $\mathcal{D} \subset \mathbb{R}^n$, pro $\boldsymbol{x} \notin \mathcal{D}$ je $f(\boldsymbol{x})$ dodefinována jako 0. Buď dále dána hustota pravděpodobnosti $g(\boldsymbol{x})$, pro kterou platí

$\mathcal{D} \subset \operatorname{supp}(g)$[3]. Pak zřejmě platí

$$\mathbb{E}_f\left(S\left(\boldsymbol{X}\right)\right) = \int_{\mathcal{D}} S\left(\boldsymbol{x}\right) \cdot f\left(\boldsymbol{x}\right) \cdot \frac{g\left(\boldsymbol{x}\right)}{g\left(\boldsymbol{x}\right)} \mathrm{d}\boldsymbol{x} = \int_{\mathcal{F}} S\left(\boldsymbol{x}\right) \cdot f\left(\boldsymbol{x}\right) \cdot \frac{g\left(\boldsymbol{x}\right)}{g\left(\boldsymbol{x}\right)} \mathrm{d}\boldsymbol{x} = \mathbb{E}_g\left(S\left(\boldsymbol{X}\right) \cdot \frac{f\left(\boldsymbol{X}\right)}{g\left(\boldsymbol{X}\right)}\right),$$

kde $\mathcal{F} \subset \mathbb{R}^n$ je definiční obor hustoty pravděpodobnosti $g\left(\boldsymbol{x}\right)$. Tento zápis umožňuje odhad $\mathbb{E}_f\left(S\left(\boldsymbol{X}\right)\right)$ za použití generace snímků z $g\left(\boldsymbol{x}\right)$ a nazývá se *importance sampling* metoda. Vlastnosti a terminologie IS metody jsou uvedeny v následující definici a větách.

Definice 3.3. Mějme funkci $S\left(\boldsymbol{x}\right) : \mathbb{R}^n \to \mathbb{R}$, hustotu pravděpodobnosti $f\left(\boldsymbol{x}\right)$ a hustotu pravděpodobnosti $g\left(\boldsymbol{x}\right)$, pro kterou platí

$$\operatorname{supp}\left(f \cdot S\right) \subset \operatorname{supp}\left(g\right), \tag{3.2}$$

v rámci IS metody nazýváme:

- $f\left(\boldsymbol{x}\right)$ nominální hustotou pravděpodobnosti,
- $g\left(\boldsymbol{x}\right)$ IS hustotou pravděpodobnosti a
- poměr $W\left(\boldsymbol{x}\right) = \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}$ *likelihood ratio* (LR) IS estimátoru.

Dále máme-li náhodný výběr $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, definujeme odhady střední hodnoty (IS estimátor)

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} S\left(\boldsymbol{X}_i\right) \cdot \frac{f\left(\boldsymbol{X}_i\right)}{g\left(\boldsymbol{X}_i\right)}$$

a směrodatné odchylky

$$s_g = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} \left(S\left(\boldsymbol{X}_i\right) \cdot \frac{f\left(\boldsymbol{X}_i\right)}{g\left(\boldsymbol{X}_i\right)} - \hat{\ell}\right)^2}. \tag{3.3}$$

Další možností, jak odhadnout střední hodnotu $\mathbb{E}_f\left(S\left(\boldsymbol{X}\right)\right)$, je použít tzv. vážený IS estimátor, jedná se o následující odhad

$$\hat{\ell}_w = \frac{\sum_{i=1}^{N} S\left(\boldsymbol{X}_i\right) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}{\sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}.$$

V případě váženého odhadu zaměňujeme počet snímků součtem *likelihood ratio*, což může v některých případech vést k efektivnějšímu odhadu. Pro velký počet snímků si jsou odhady téměř rovny, neboť střední hodnota *likelihood ratio* je rovná jedné, viz následující věta a důkaz.

[3] supportem funkce rozumíme množinu $\operatorname{supp}\left(g\right) = \overline{\left\{\boldsymbol{x} \in \mathbb{R}^n : g\left(\boldsymbol{x}\right) \neq 0\right\}}$

Věta 3.4 (Střední hodnota *likelihood ratio*)**.** *Nechť $f(\boldsymbol{x})$ je hustotu pravděpodobnosti a $g(\boldsymbol{x})$ hustotu pravděpodobnosti splňující (3.2), pak pro střední hodnotu likelihood ratio platí*

$$\mathbb{E}_g\left(\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right) = 1.$$

Důkaz. Zapišme střední hodnotu v integrálním tvaru

$$\mathbb{E}_g\left(\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right) = \int_{\mathcal{F}} \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \cdot g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{F}} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1,$$

kde $\mathcal{F} \subset \mathbb{R}^n$ je definiční obor hustoty pravděpodobnosti $g(\boldsymbol{x})$ a funkce $f(\boldsymbol{x})$ je nenulová pouze na $\mathcal{D} \subset \mathcal{F}$. □

Věta 3.5 (Nestranost IS estimátorů)**.** *Nechť $S(\boldsymbol{x})$ je funkcí v $\mathbb{R}^n$, $f(\boldsymbol{x})$ je hustotou pravděpodobnosti a $g(\boldsymbol{x})$ je hustotou pravděpodobnosti, pro niž platí (3.2). Pak odhady*

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} S(\boldsymbol{X}_i) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)} \;\; a \;\; \hat{\ell}_w = \frac{\sum_{i=1}^{N} S(\boldsymbol{X}_i) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}{\sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}$$

jsou nestrannými odhady $\mathbb{E}_f(S(\boldsymbol{X}))$.

Důkaz. Střední hodnotu odhadu $\hat{\ell}$ můžeme zapsat jako

$$\begin{aligned}\mathbb{E}_g\left(\hat{\ell}\right) &= \mathbb{E}_g\left(\frac{1}{N}\sum_{i=1}^{N} S(\boldsymbol{X}_i) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}\right) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_g\left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_f(S(\boldsymbol{X})) = \mathbb{E}_f(S(\boldsymbol{X})).\end{aligned}$$

Střední hodnotu odhadu $\hat{\ell}_w$,můžeme zapsat jako

$$\begin{aligned}\mathbb{E}_g\left(\hat{\ell}_w\right) &= \mathbb{E}_g\left(\frac{\sum_{i=1}^{N} S(\boldsymbol{X}_i) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}{\sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}\right) = \frac{\mathbb{E}_g\left(\sum_{i=1}^{N} S(\boldsymbol{X}_i) \cdot \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}\right)}{\mathbb{E}_g\left(\sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}\right)} \\ &= \frac{\sum_{i=1}^{N} \mathbb{E}_g\left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right)}{\sum_{i=1}^{N} \mathbb{E}_g\left(\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right)} = \frac{\sum_{i=1}^{N} \mathbb{E}_f(S(\boldsymbol{X}))}{N} = \mathbb{E}_f(S(\boldsymbol{X})).\end{aligned}$$

Čímž jsme dokázali tvrzení. □

IS metoda se dá shrnout do následujícího algoritmu.

Algoritmus 3.6 (Importance sampling metoda)**.**

Předpoklady: hustota pravděpodobnosti $g(\boldsymbol{x})$, pro kterou platí (3.2).

1. Vygenerování náhodného výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ z rozdělení pravděpodobnosti daného hustotou pravděpodobnosti $g(\boldsymbol{x})$.
2. Výpočet odhadu $\mathbb{E}_f(S(\boldsymbol{X}))$ pomocí estimátoru
$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} S(\boldsymbol{X}_i)\cdot\frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}, \text{ nebo } \hat{\ell}_w = \frac{\sum_{i=1}^{N} S(\boldsymbol{X}_i)\cdot\frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}{\sum_{i=1}^{N}\frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)}}.$$
3. Sestavení $1-\alpha$ konfidenčního intervalu
$$\left(\hat{\ell} - \frac{z_{1-\alpha/2}\cdot s_g}{\sqrt{N}}, \hat{\ell} + \frac{z_{1-\alpha/2}\cdot s_g}{\sqrt{N}}\right),$$
kde $z_{1-\alpha/2}$ značí $1-\alpha/2$ kvantil normovaného normálního rozdělení a s_g je výběrová směrodatné odchylka daná (3.3).

Nyní je třeba určit vhodnou IS hustotu pravděpodobnosti, tak aby měl odhad pomocí IS menší rozptyl, než odhad původní.

3.1.1 IS hustota pravděpodobnosti s minimálním rozptylem

Jelikož je cílem získat co nejmenší rozptyl IS estimátoru $S(\boldsymbol{X})\cdot\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}$, nabízí se volit IS hustotu pravděpodobnosti jako řešení úlohy
$$\operatorname*{argmin}_g \operatorname{Var}_g\left(S(\boldsymbol{X})\cdot\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right).$$
Pro specifický případ, kdy je $S(\boldsymbol{x})$ pouze kladná, nebo záporná, je řešení jednoduché, a to
$$g(\boldsymbol{x}) = \frac{S(\boldsymbol{x})\cdot f(\boldsymbol{x})}{\ell}.$$
V tomto případě získáme nulový rozptyl
$$\operatorname{Var}_g\left(S(\boldsymbol{X})\cdot\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}\right) = \int_{\mathcal{F}}\left(\frac{S(\boldsymbol{x})\cdot f(\boldsymbol{x})}{\frac{S(\boldsymbol{x})\cdot f(\boldsymbol{x})}{\ell}} - \ell\right)^2\cdot\frac{S(\boldsymbol{x})\cdot f(\boldsymbol{x})}{\ell}\,\mathrm{d}\boldsymbol{x} = 0.$$
Na obecný případ, kdy funkce $S(\boldsymbol{x})$ mění své znaménko, odpoví následující věta.

Věta 3.7 (Optimální IS hustota pravděpodobnosti)**.** *Nechť $S(\boldsymbol{x})$ je funkcí v $\mathbb{R}^n$ a $f(\boldsymbol{x})$ je hustotou pravděpodobnosti, pak má úloha*

$$\underset{g}{\operatorname{argmin}} \operatorname{Var}_g \left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right)$$

jediné řešení ve tvaru

$$g^*(\boldsymbol{x}) = \frac{|S(\boldsymbol{x})| \cdot f(\boldsymbol{x})}{\mathbb{E}_f(|S(\boldsymbol{x})|)}.$$

(Uvažujeme pouze případy, kdy $\mathbb{E}_f(|S(\boldsymbol{x})|) \neq 0$, v opačném případě by již klasická MC metoda dosahovala nulového rozptylu.)

Důkaz. Pro důkaz tvrzení využijeme alternativní vzorec pro rozptyl

$$\operatorname{Var}_f(\boldsymbol{X}) = \mathbb{E}_f\left(\boldsymbol{X}^2\right) - \mathbb{E}_f(\boldsymbol{X})^2$$

a nerovnost

$$\mathbb{E}_f\left(\boldsymbol{X}^2\right) \geq \mathbb{E}_f(\boldsymbol{X})^2.$$

Jelikož

$$\mathbb{E}_g \left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right)^2 = \ell^2$$

je konstanta pro libovolnou hustotu pravděpodobnosti $g(\boldsymbol{x})$ splňující (3.2), stačí ukázat, že $g^*(\boldsymbol{x})$ je řešením minimalizační úlohy

$$\underset{g}{\operatorname{argmin}} \mathbb{E}_g \left(\left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right)^2 \right).$$

Pro libovolnou hustotu pravděpodobnosti splňující (3.2) tedy platí:

$$\begin{aligned}
\mathbb{E}_g \left(\left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right)^2 \right) &\geq \mathbb{E}_g \left(S(\boldsymbol{X}) \cdot \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right)^2 = \mathbb{E}_f(S(\boldsymbol{X}))^2 \\
&= \left(\int_{\mathcal{F}} S(\boldsymbol{x}) \cdot f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right)^2 = \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot f(\boldsymbol{x})^2 \, \mathrm{d}\boldsymbol{x} \\
&= \frac{\mathbb{E}_f(|S(\boldsymbol{X})|)}{\mathbb{E}_f(|S(\boldsymbol{X})|)} \cdot \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot f(\boldsymbol{x})^2 \, \mathrm{d}\boldsymbol{x} \\
&= \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot \frac{f(\boldsymbol{x})^2}{\frac{|S(\boldsymbol{x})| \cdot f(\boldsymbol{x})}{\mathbb{E}_f(|S(\boldsymbol{X})|)}} \mathrm{d}\boldsymbol{x} = \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot \frac{f(\boldsymbol{x})^2}{g^*(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_g \left(\left(|S(\boldsymbol{X})| \cdot \frac{f(\boldsymbol{X})}{g^*(\boldsymbol{X})} \right)^2 \right).
\end{aligned}$$

Čímž je tvrzení dokázáno. □

Důsledek 3.8. *Všimněme si, že předchozí důkaz mimo jiné udává minimální možný rozptyl dosažitelný pomocí IS metody*

$$\mathrm{Var}_{g^*}\left(S\left(\boldsymbol{X}\right)\cdot\frac{f\left(\boldsymbol{X}\right)}{g^*\left(\boldsymbol{X}\right)}\right)=\mathbb{E}_f\left(\left|S\left(\boldsymbol{X}\right)\right|\right)^2-\mathbb{E}_f\left(S\left(\boldsymbol{X}\right)\right)^2.$$

Nyní ukažme aplikaci IS hustoty s minimálním rozptylem na příklad 3.1.

Příklad 3.9. Najděme hustotu pravděpodobnosti pro docílení minimálního rozptylu IS estimátoru v případě, že rozdělení pravděpodobnosti je $\mathcal{N}\left(0,1\right)$ a zkoumaná funkce $S\left(x\right)=I\left(x>2\right)$. Použijeme konstrukci hustoty pravděpodobnosti z věty 3.7:

$$g^*(x)=\frac{I\left(x>2\right)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)}{1-F\left(2\right)}=\begin{cases}0, & x<2\\ \frac{1}{\left(1-F(2)\right)\cdot\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right), & x\geq 2\end{cases}.$$

Jelikož je $S\left(x\right)\geq 0$ získáváme IS estimátor s nulovým rozptylem. Jednotlivé snímky mají tedy tvar

$$I\left(x>2\right)\cdot\frac{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)}{\frac{I(x>2)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)}{1-F(2)}}=1-F\left(2\right).$$

Snímky, pro které $I\left(x>2\right)=0$, nemusíme brát v úvahu, neboť podle hustoty $g^*\left(x\right)$ vůbec nenastanou. △

V praxi je však tato optimální hustota pravděpodobnosti nepoužitelná, neboť její konstrukce vyžaduje znalost řešení původní úlohy. Je to však užitečná znalost, která se dá využít při odvození jiných metod pro určení IS hustoty pravděpodobnosti.

3.2 Metoda minimalizace rozptylu

Před odvozením metody je třeba nejprve uvést následující definici.

Definice 3.10 (Parametrická rodina pravděpodobnostních rozdělení)**.** Mějme hustotu rozdělení $f\left(\boldsymbol{x};\boldsymbol{v}\right)$, kde $\boldsymbol{v}$ je parametr rozdělení. Mějme dále Θ, množinu všech možných parametrů $\boldsymbol{v}$, pro které je $f\left(\boldsymbol{x};\boldsymbol{v}\right)$ hustotou pravděpodobnosti a $\mathrm{supp}\, f\left(\boldsymbol{x};\boldsymbol{v}\right)$ se nemění s měnícím se $\boldsymbol{v}\in\Theta$. Potom množinu

$$\mathbb{V}=\left\{f\left(\cdot;\boldsymbol{v}\right),\boldsymbol{v}\in\Theta\right\}$$

budeme nazývat *parametrická rodina pravděpodobnostních rozdělení*.

V předchozí části byla odvozena optimální IS hustota pravděpodobnosti, která však nemá praktické využití. Možností, jak nalézt použitelnou IS hustotu pravděpodobnosti, je omezit množinu možných hustot pravděpodobností, přes kterou minimalizujeme

$$\underset{g\in\mathbb{V}}{\mathrm{argmin}}\,\mathrm{Var}_g\left(S\left(\boldsymbol{X}\right)\cdot\frac{f\left(\boldsymbol{X}\right)}{g\left(\boldsymbol{X}\right)}\right).$$

Pokud za tuto množinu zvolíme parametrickou rodinu pravděpodobnostních rozdělení, do níž patří $f(\boldsymbol{x})$

$$\mathbb{V} = \{f(\cdot;\boldsymbol{v}), \boldsymbol{v} \in \Theta\},$$

kde $\boldsymbol{v}$ je parametrem nominální hustoty pravděpodobnosti $f(\boldsymbol{x})$ a Θ je množinou všech přípustných parametrů, získáme metodu minimalizace rozptylu (*„variance minimalization method"* dále jen VM). V tomto případě budeme psát *likelihood ratio* jako

$$W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v}) = \frac{f(\boldsymbol{X};\boldsymbol{u})}{f(\boldsymbol{X};\boldsymbol{v})}.$$

Řešíme tedy problém

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta} \operatorname{Var}_{\boldsymbol{v}}(S(\boldsymbol{X}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v})),$$

kde $\boldsymbol{u}$ je parametr definující nominální hustotu pravděpodobnosti. Při využití vztahu $\operatorname{Var}_f(\boldsymbol{X}) = \mathbb{E}_f(\boldsymbol{X}^2) - \mathbb{E}_f(\boldsymbol{X})^2$ získáme ekvivalentní definici problému

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta} \mathbb{E}_{\boldsymbol{v}}\left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v})^2\right).$$

V případě numerické aproximace by tento vztah nebyl vhodný, neboť neznáme parametr $\boldsymbol{v}$ a tudíž nejsme schopni generovat snímky. Proto využijeme vztahu

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v}}\left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v})^2\right) &= \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot W(\boldsymbol{x};\boldsymbol{u},\boldsymbol{v}) \cdot \frac{f(\boldsymbol{x};\boldsymbol{u})}{f(\boldsymbol{x};\boldsymbol{v})} \cdot f(\boldsymbol{x};\boldsymbol{v})\,\mathrm{d}\boldsymbol{x} \\
&= \int_{\mathcal{F}} S(\boldsymbol{x})^2 \cdot W(\boldsymbol{x};\boldsymbol{u},\boldsymbol{v}) \cdot \frac{f(\boldsymbol{x};\boldsymbol{u})}{f(\boldsymbol{x};\boldsymbol{w})} \cdot f(\boldsymbol{x};\boldsymbol{w})\,\mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{\boldsymbol{w}}\left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})\right),
\end{aligned}$$

který umožní ekvivalentní definici VM problému

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta} \mathbb{E}_{\boldsymbol{w}}\left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})\right),$$

případně

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta} \mathbb{E}_{\boldsymbol{u}}\left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v})\right).$$

Pro možnost numerického řešení definujme následující metodu.

Definice 3.11 (*„Stochastic counterpart"*)**.** Mějme problém

$$\boldsymbol{v}^* = \operatorname*{argmin}_{\boldsymbol{v}\in\Theta} \mathbb{E}_{\boldsymbol{u}}(H(\boldsymbol{X};\boldsymbol{v})), \text{ nebo } \boldsymbol{v}^* = \operatorname*{argmax}_{\boldsymbol{v}\in\Theta} \mathbb{E}_{\boldsymbol{u}}(H(\boldsymbol{X};\boldsymbol{v})),$$

kde $H(\boldsymbol{X};\boldsymbol{v})$ je funkce náhodné veličiny $\boldsymbol{X} \sim f(\boldsymbol{x};\boldsymbol{u})$ a hledaného parametru $\boldsymbol{v}$. Pak přibližné řešení můžeme získat jako

$$\widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmin}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} H(\boldsymbol{X}_i;\boldsymbol{v}) \right), \text{nebo } \widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} H(\boldsymbol{X}_i;\boldsymbol{v}) \right).$$

Tento postup nazýváme *„stochastic counterpart" (SC)* řešeného problému.

Poznámka 3.12. Dá se dokázat (viz [3]), že řešení SC konverguje s $N \to \infty$ ke skutečnému řešení, tedy

$$N \to \infty : \mathbb{P}\left(\boldsymbol{v}^* = \widetilde{\boldsymbol{v}^*}\right) = 1.$$

Nyní shrňme poznatky o VM metodě.

Definice 3.13 (Metoda minimalizace rozptylu)**.** Mějme parametrickou rodinu pravděpodobnostních rozdělení

$$\mathbb{V} = \{f(\cdot;\boldsymbol{v}), \boldsymbol{v} \in \Theta\}$$

a úlohu na určení $\mathbb{E}_{\boldsymbol{u}}(S(\boldsymbol{X}))$. Metodou minimalizace rozptylu rozumíme hledání optimálního parametru $\boldsymbol{v}^* \in \Theta$ jako

$$\boldsymbol{v}^* = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmin}} \mathbb{E}_{\boldsymbol{w}} \left(S(\boldsymbol{X})^2 \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}) \right).$$

Aproximaci řešení VM problému získáme pomocí SC jako

$$\widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmin}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(S(\boldsymbol{X}_i)^2 \cdot W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{v}) \cdot W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{w}) \right) \right),$$

kde $\boldsymbol{X} \sim f(\cdot;\boldsymbol{w})$.

Všimněme si, že při řešení SC máme možnost generovat snímky z libovolného rozdělení $f(\cdot;\boldsymbol{w})$. Toto nám umožňuje použití iteračního řešení VM úlohy.

Algoritmus 3.14 (VM algoritmus)**.**

1. Volba počátečního $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}$, $t = 1$.
2. Vygenerování náhodného výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim f(\cdot;\widetilde{\boldsymbol{v}}_{t-1})$.

3. Aktualizace parametru

$$\widetilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmin}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(S\left(\boldsymbol{X}_i\right)^2 \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \boldsymbol{v}\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right) \right) \right).$$

4. Pokud je splněna ukončovací podmínka (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1} \approx \widetilde{\boldsymbol{v}}_t$, ...), ukončit, jinak zpět ke kroku 2.

Ukažme výsledky aplikace VM metody na motivační příklad 3.1.

Příklad 3.15. Najděme hustotu pravděpodobnosti pro docílení minimálního rozptylu IS estimátoru, v případě, že rozdělení pravděpodobnosti je $\mathcal{N}(0,1)$, zkoumaná funkce $S(x) = I(x > 2)$ a parametrickou rodinou pravděpodobnostních rozdělení v které hledáme optimální hustotu pravděpodobnosti je

$$\mathbb{V} = \left\{ \forall v \in \mathbb{R} : \mathcal{N}(v, 1) \right\}.$$

V případě analytické formule VM problému musíme vyřešit následující úlohu:

$$\begin{aligned}
v^* &= \underset{v\in\mathbb{R}}{\operatorname{argmin}} \mathbb{E}_0 \left(I\left(x > 2\right)^2 \cdot W\left(\boldsymbol{X}; 0, v\right) \right) = \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \frac{\left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-0)^2}{2} \right) \right)^2}{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-v)^2}{2} \right)} \mathrm{d}x \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-0)^2}{2} \right) \right)^2 \cdot \sqrt{2\pi} \cdot \exp\left(\frac{(x-v)^2}{2} \right) \mathrm{d}x \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \frac{1}{\sqrt{2\pi}} \cdot \exp\left(\frac{(x-v)^2}{2} - x^2 \right) \mathrm{d}x = \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \exp\left(\frac{(x-v)^2}{2} - x^2 \right) \mathrm{d}x.
\end{aligned}$$

Pro vyřešení této úlohy byl použit Matlab a výsledkem je $v^* \doteq 2.2159$. Optimální hustota pravděpodobnosti IS estimátoru patřící do $\mathbb{V}$ je $\mathcal{N}(2.2159, 1)$.

Dále vyřešíme VM problém této úlohy pomocí SC (opět pomocí Matlabu). Pro toto řešení použijeme algoritmus 3.14 s parametry $N = 20$ a pevným počtem kroků $K = 5$. Výsledná aproximace optimálního parametru je $\widetilde{v^*} \doteq 2.2374$. Aproximace optimální hustoty pravděpodobnosti IS estimátoru patřící do $\mathbb{V}$ je $\mathcal{N}(2.2374, 1)$. △

Jak bylo možno vidět při řešení příkladu 3.15, řešení VM problému vyžaduje složitý matematický aparát. I v případě využití SC VM problému je třeba řešit netriviální minimalizační problém v každé iteraci algoritmu. Proto je nutné nalezení metody, která bude poskytovat snadnější cestu k „optimální" IS hustotě pravděpodobnosti.

3.3 Cross-entropy metoda

Tato část čerpá z [1, 7]. Pro získání dobré IS hustoty pravděpodobnosti lze použít kromně minimalizace rozptylu i jiný přístup. Jelikož již známe konstrukci optimální IS hustoty pravděpodobnosti, můžeme minimalizovat vzdálenost k tomuto rozdělení. Nejprve však musíme nalézt vhodnou míru vzdálenosti.

3.3.1 Kullback-Leiblerova divergence

Jednou z možností jak měřit „vzdálenost" (nejedná se formálně o vzdálenost) mezi hustotami pravděpodobnosti je Kullback-Leiblerova divergence (také nazývána cross-entropy).

Definice 3.16 (Kullback-Leiblerova divergence (cross-entropy))**.** Mějme g a h hustoty pravděpodobnosti definované na množině $\mathcal{X}$, pro které platí

$$\operatorname{supp} g \subset \operatorname{supp} h. \tag{3.4}$$

Kullback-Leiblerovou divergencí nazýváme

$$\mathcal{D}(g,h) = \mathbb{E}_g\left(\ln \frac{g(\boldsymbol{X})}{h(\boldsymbol{X})}\right).$$

Což lze vyjádřit jako

$$\mathcal{D}(g,h) = \int_{\mathcal{X}} g(\boldsymbol{x}) \cdot \ln \frac{g(\boldsymbol{x})}{h(\boldsymbol{x})} \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{X}} g(\boldsymbol{x}) \cdot \ln g(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{X}} g(\boldsymbol{x}) \cdot \ln h(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}.$$

Jak už bylo dříve zmíněno nejedná se skutečně o vzdálenost, neboť obecně

$$\mathcal{D}(g,h) \neq \mathcal{D}(h,g).$$

Avšak jiné vlastnosti vzdálenosti splňuje, což často vede k označování K-L divergence jako vzdálenosti.

Věta 3.17. *Nechť g, h jsou libovolné hustoty pravděpodobnosti definované na množině $\mathcal{X}$ splňující (3.4). Pak pro Kullback-Leiblerovu divergenci platí*

1. *K-L divergence je nezáporná:*
$$\mathcal{D}(g,h) \geq 0.$$
2. *Platí ekvivalence*
$$(g(\boldsymbol{x}) = h(\boldsymbol{x}) \text{ skoro všude v } \mathcal{X}) \Leftrightarrow \mathcal{D}(g,h) = 0.$$
Termín skoro všude znamená až na podmnožinu $\mathcal{X}$, jejíž míra je nulová.

Důkaz.

1. Pro toto tvrzení použijeme Jensenovu nerovnost:
$$\phi \text{ je konvexní funkce } \Rightarrow \mathbb{E}(\phi(\boldsymbol{X})) \geq \phi(\mathbb{E}(\boldsymbol{X}))$$

a vlastnost

$$\mathbb{E}_g\left(\frac{h\left(\boldsymbol{X}\right)}{g\left(\boldsymbol{X}\right)}\right)=\int\limits_{\mathcal{X}}\frac{h\left(\boldsymbol{x}\right)}{g\left(\boldsymbol{x}\right)}\cdot g\left(\boldsymbol{x}\right)\mathrm{d}\boldsymbol{x}=\int\limits_{\mathcal{X}}h\left(\boldsymbol{x}\right)\mathrm{d}\boldsymbol{x}=1.$$

Pak se znalostí, že $-\ln$ je konvexní funkce, můžeme psát

$$\mathcal{D}\left(g,h\right)=\mathbb{E}_g\left(\ln\frac{g\left(\boldsymbol{X}\right)}{h\left(\boldsymbol{X}\right)}\right)=\mathbb{E}_g\left(-\ln\frac{h\left(\boldsymbol{X}\right)}{g\left(\boldsymbol{X}\right)}\right)\geq-\ln\mathbb{E}_g\left(\frac{h\left(\boldsymbol{X}\right)}{g\left(\boldsymbol{X}\right)}\right)=-\ln 1=0.$$

Čímž jsme dokázali první tvrzení.

2. Vyplývá z výše zmíněné Jensenovy nerovnosti. Jelikož je $-\ln$ striktně konvexní funkce, pak se z Jensenovy nerovnosti stává rovnost právě tehdy když jsou si $h\left(\boldsymbol{x}\right)$ a $g\left(\boldsymbol{x}\right)$ rovny skoro všude.

□

3.3.2 Cross-entropy metoda pro estimace

Mějme nyní opět úlohu na hledání optimální hustoty pravděpodobnosti z nějaké zadané parametrické rodiny pravděpodobnostních rozdělení $\mathbb{V}$. Jak bylo dříve zmíněno, využijeme znalost konstrukce hustoty pravděpodobnosti s minimálním rozptylem g^* (viz 3.7) a budeme hledat prvek $\mathbb{V}$ s minimální vzdáleností od g^*. Tedy budeme řešit problém

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta}\operatorname{dist}\left(g^*,f\left(\cdot;\boldsymbol{v}\right)\right),$$

přičemž jako míru vzdálenosti využijeme právě K-L divergenci

$$\operatorname*{argmin}_{\boldsymbol{v}\in\Theta}\mathcal{D}\left(g^*,f\left(\cdot;\boldsymbol{v}\right)\right).$$

Dále upravme minimalizační úlohu

$$\begin{aligned}\boldsymbol{v}^* &= \operatorname*{argmin}_{\boldsymbol{v}\in\Theta}\mathcal{D}\left(g^*,f\left(\cdot;\boldsymbol{v}\right)\right)=\operatorname*{argmin}_{\boldsymbol{v}\in\Theta}\left(\underbrace{\int\limits_{\mathcal{X}}g^*\left(\boldsymbol{x}\right)\ln g^*\left(\boldsymbol{x}\right)\mathrm{d}\boldsymbol{x}}_{\text{konstanta}}-\int\limits_{\mathcal{X}}g\left(\boldsymbol{x}\right)\ln f\left(\boldsymbol{x};\boldsymbol{v}\right)\mathrm{d}\boldsymbol{x}\right)\\ &= \operatorname*{argmin}_{\boldsymbol{v}\in\Theta}\left(-\int\limits_{\mathcal{X}}g\left(\boldsymbol{x}\right)^*\cdot\ln f\left(\boldsymbol{x};\boldsymbol{v}\right)\mathrm{d}\boldsymbol{x}\right)=\operatorname*{argmax}_{\boldsymbol{v}\in\Theta}\left(\int\limits_{\mathcal{X}}g\left(\boldsymbol{x}\right)^*\cdot\ln f\left(\boldsymbol{x};\boldsymbol{v}\right)\mathrm{d}\boldsymbol{x}\right).\end{aligned}$$

Tato úloha je však stále neřešitelná, neboť obsahuje neznámou funkci g^*. Tohoto problému se však lze zbavit následující úpravou

$$\begin{aligned}
\boldsymbol{v}^* &= \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\int_{\mathcal{X}} g\left(\boldsymbol{x}\right)^* \ln f\left(\boldsymbol{x};\boldsymbol{v}\right) \mathrm{d}\boldsymbol{x} \right) = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\int_{\mathcal{X}} \frac{\left|S\left(\boldsymbol{x}\right)\right| \cdot f\left(\boldsymbol{x};\boldsymbol{u}\right)}{\mathbb{E}_{\boldsymbol{u}}\left(\left|S\left(\boldsymbol{X}\right)\right|\right)} \ln f\left(\boldsymbol{x};\boldsymbol{v}\right) \mathrm{d}\boldsymbol{x} \right) \\
&= \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \underbrace{\frac{1}{\mathbb{E}_{\boldsymbol{u}}\left(\left|S\left(\boldsymbol{X}\right)\right|\right)}}_{\text{konstanta}} \cdot \left(\int_{\mathcal{X}} \left|S\left(\boldsymbol{x}\right)\right| \cdot f\left(\boldsymbol{x};\boldsymbol{u}\right) \cdot \ln f\left(\boldsymbol{x};\boldsymbol{v}\right) \mathrm{d}\boldsymbol{x} \right) \\
&= \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\int_{\mathcal{X}} \left|S\left(\boldsymbol{x}\right)\right| \cdot f\left(\boldsymbol{x};\boldsymbol{u}\right) \cdot \ln f\left(\boldsymbol{x};\boldsymbol{v}\right) \mathrm{d}\boldsymbol{x} \right) = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{u}}\left(\left|S\left(\boldsymbol{X}\right)\right| \cdot \ln f\left(\boldsymbol{X};\boldsymbol{v}\right)\right).
\end{aligned}$$

Pro numerické řešení bude velice užitečná následující úprava

$$\boldsymbol{v}^* = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{u}}\left(\left|S\left(\boldsymbol{X}\right)\right| \cdot \ln f\left(\boldsymbol{X};\boldsymbol{v}\right)\right) = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{w}}\left(\left|S\left(\boldsymbol{X}\right)\right| \cdot \ln f\left(\boldsymbol{X};\boldsymbol{v}\right) \cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right),$$

která umožní při numerické aproximaci pomocí SC využít libovolné rozdělení z $\mathbb{V}$.

Dále shrňme poznatky o CE metodě do následující definice.

Definice 3.18 (Cross-entropy metoda)**.** Mějme parametrickou rodinu pravděpodobnostních rozdělení

$$\mathbb{V} = \left\{ f\left(\cdot;\boldsymbol{v}\right), \boldsymbol{v} \in \Theta \right\}$$

a úlohu na určení $\mathbb{E}_{\boldsymbol{u}}\left(S\left(\boldsymbol{X}\right)\right)$. Cross-entropy metodou rozumíme hledání optimálního parametru $\boldsymbol{v}^* \in \Theta$ jako

$$\boldsymbol{v}^* = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{w}}\left(\left|S\left(\boldsymbol{X}\right)\right| \cdot \ln f\left(\boldsymbol{X};\boldsymbol{v}\right) \cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right).$$

Aproximaci řešení CE úlohy získáme pomocí SC jako

$$\widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(\left|S\left(\boldsymbol{X}_i\right)\right| \cdot \ln f\left(\boldsymbol{X}_i;\boldsymbol{v}\right) \cdot W\left(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{w}\right)\right) \right),$$

kde $\boldsymbol{X} \sim f\left(\boldsymbol{x};\boldsymbol{w}\right)$.

Stejně jako v případě VM metody (algoritmus 3.14) můžeme sestavit iterační algoritmus pro numerické řešení CE úlohy.

Algoritmus 3.19 (CE algoritmus)**.**

1. Volba počátečního $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}$, $t = 1$.

2. Vygenerování náhodného výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim f\left(\cdot; \widetilde{\boldsymbol{v}}_{t-1}\right)$.

3. Aktualizace parametru

$$\widetilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(\left| S\left(\boldsymbol{X}_i\right) \right| \cdot \ln f\left(\boldsymbol{X}_i; \boldsymbol{v}\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right) \right) \right). \tag{3.5}$$

4. Pokud je splněna ukončovací podmínka (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1} \approx \widetilde{\boldsymbol{v}}_t$, ...), ukončit, jinak zpět ke kroku 2.

Je důležité zmínit, že oproti VM metodě má v případě rozdělení z exponenciální rodiny krok 3. poměrně často jednoduché analytické řešení. Přičemž v případě VM metody jsme mnohdy nuceni využít numerického řešení.

Dále uveďme CE řešení motivační úlohy.

Příklad 3.20. Najděme hustotu pravděpodobnosti pro docílení minimálního rozptylu IS estimátoru, v případě, že rozdělení pravděpodobnosti je $\mathcal{N}\left(0,1\right)$, zkoumaná funkce $S\left(x\right) = I\left(x > 2\right)$ a parametrickou rodinou pravděpodobnostních rozdělení, v které hledáme optimální hustotu pravděpodobnosti, je

$$\mathbb{V} = \left\{ \forall v \in \mathbb{R} : \mathcal{N}\left(v, 1\right) \right\}.$$

V případě analytické formule CE úlohy řešíme následující optimalizační problém:

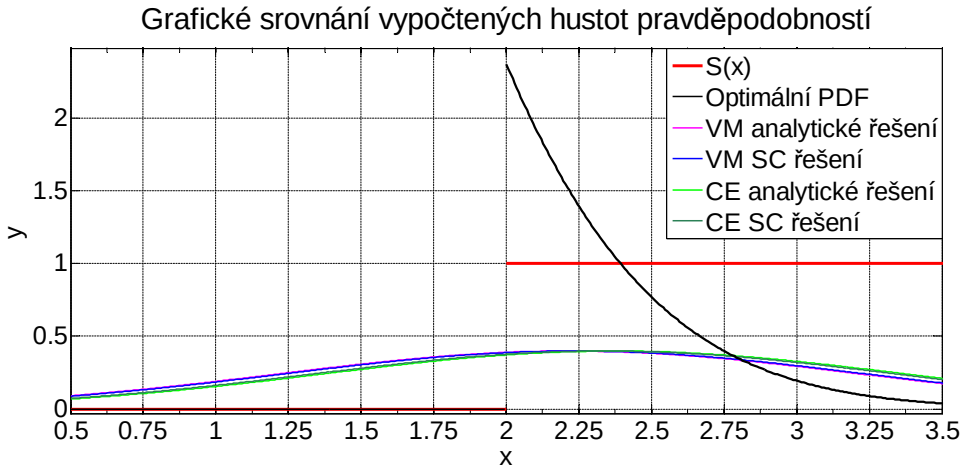
$$\begin{aligned}
v^* &= \underset{v\in\mathbb{R}}{\operatorname{argmax}}\, \mathbb{E}_0 \left(I\left(x > 2\right)^2 \cdot \ln f\left(\boldsymbol{X}; v\right) \right) \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmax}} \int_2^\infty \left(-\frac{1}{2}\ln 2\pi - \frac{\left(x-v\right)^2}{2} \right) \cdot \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(x-0\right)^2}{2} \right) \right) \mathrm{d}x \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmax}} \int_2^\infty -\frac{\left(x-v\right)^2}{2} \cdot \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2} \right) \right) \mathrm{d}x = \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \left(x-v\right)^2 \cdot \exp\left(-\frac{x^2}{2} \right) \mathrm{d}x = \underset{v\in\mathbb{R}}{\operatorname{argmin}} \int_2^\infty \left(v^2 - 2vx\right) \cdot \exp\left(-\frac{x^2}{2} \right) \mathrm{d}x \\
&= \underset{v\in\mathbb{R}}{\operatorname{argmin}} v^2 \int_2^\infty \exp\left(-\frac{x^2}{2} \right) \mathrm{d}x - 2v \cdot \int_2^\infty x \cdot \exp\left(-\frac{x^2}{2} \right) \mathrm{d}x,
\end{aligned}$$

což je pouze minimalizace kvadratické funkce. Pro vyřešení této úlohy byl použit Matlab a výsledkem je $v^* \doteq 2.3732$. Optimální hustota pravděpodobnosti IS estimátoru patřící do $\mathbb{V}$ je $\mathcal{N}\left(2.3732, 1\right)$.

Dále vyřešíme CE úlohu pomocí SC. Pro toto řešení použijeme algoritmus 3.19 s parametry $N = 20$ a pevným počtem kroků $K = 5$. Výsledná aproximace optimálního parametru je $\widetilde{v^*} \doteq 2.3382$. Aproximace optimální hustoty pravděpodobnosti IS estimátoru patřící do $\mathbb{V}$ je $\mathcal{N}\left(2.3382, 1\right)$. △

Jak lze vidět, v případě CE metody je řešení mnohem jednodušší než v případě VM metody. Při řešení CE metodou stačí v každém kroku najít minimum kvadratické funkce, což umožňuje efektivní PC implementaci. Dále v této kapitole ukážeme úpravu CE metody, která pro vybrané typy pravděpodobnostních rozdělení zjednoduší řešení ještě více. Nejprve však uveďme srovnání řešení motivačního příkladu.

Příklad 3.21 (Srovnání řešení motivační úlohy)**.** Nejprve pro názornost, jak jsou si jednotlivá řešení VM a CE blízká, uveďme graf nalezených optimálních hustot pravděpodobností.



Obrázek 3.2: Motivační úloha: srovnání nalezených hustot pravděpodobností

Pro přesnější srovnání efektivity metod uveďme tabulku analyticky spočtených rozptylů IS estimátoru při použití jednotlivých metod a výslednou redukci rozptylu oproti CMC estimátoru.

Metoda	VM analyticky	VM SC	CE analyticky	CE SC
Rozptyl	$1.1700 \cdot 10^{-3}$	$1.1704 \cdot 10^{-3}$	$1.1919 \cdot 10^{-3}$	$1.1832 \cdot 10^{-3}$
Redukce	19.0016	18.9950	18.6526	18.7896

Tabulka 3.1: Motivační úloha: rozptyl IS estimátoru při použití jednotlivých PDF

Jak lze vidět z výsledných rozptylů rozdíl mezi jednotlivými metodami a jejich SC aproximacemi je minimální. Tedy nejefektivnějším způsobem hledání IS hustoty pravděpodobnosti vzhledem k poměru redukce rozptylu a náročnosti výpočtu je CE metoda řešená SC přístupem. △

Další zjednodušení výpočtu CE problému umožňuje fakt, že v běžných aplikacích je funkce

$$\phi(\boldsymbol{v}) = \mathbb{E}_{\boldsymbol{w}}\left(|S(\boldsymbol{X})| \cdot \ln f(\boldsymbol{X};\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})\right) \tag{3.6}$$

konkávní a tedy má pouze jedno maximum. Což za předpokladu diferencovatelnosti $\phi(\boldsymbol{v})$ umožňuje ekvivalentní formulaci

$$\boldsymbol{v}^* \text{ je řešením: } \nabla_{\boldsymbol{v}} \mathbb{E}_{\boldsymbol{w}} \left(|S(\boldsymbol{X})| \cdot \ln f(\boldsymbol{X}; \boldsymbol{v}) \cdot W(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}) \right) = 0.$$

Dále můžeme za předpokladu, že $\ln f(\boldsymbol{X}; \boldsymbol{v})$ je všude diferencovatelná podle $\boldsymbol{v}$, upravit na

$$\boldsymbol{v}^* \text{ je řešením: } \mathbb{E}_{\boldsymbol{w}} \left(|S(\boldsymbol{X})| \cdot \nabla_{\boldsymbol{v}} \ln f(\boldsymbol{X}; \boldsymbol{v}) \cdot W(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}) \right) = 0.$$

Přičemž záměnu derivace a integrálu můžeme ve spojitém případě zdůvodnit aplikací Lebesgueovy věty o záměně limity a integrálu (viz [15]).

Potom lze formulovat řešení SC CE úlohy jako

$$\widetilde{\boldsymbol{v}^*} \text{ je řešením soustavy rovnic: } \frac{1}{N} \cdot \sum_{i=1}^{N} \left(|S(\boldsymbol{X}_i)| \cdot \nabla \ln f(\boldsymbol{X}_i; \boldsymbol{v}) \cdot W(\boldsymbol{X}_i; \boldsymbol{u}, \boldsymbol{w}) \right) = 0.$$

V případě *„natural exponencial family"* (dále jen NEF) parametrizované střední hodnotou se díky předchozí upravě zbavíme v kroku 3. algoritmu 3.19 nutnosti řešení jakýchkoliv rovnic.

Definice 3.22 (NEF parametrizované střední hodnotou)**.** Dané parametrické rodině pravděpodobnostních rozdělení budeme říkat NEF parametrizovaná střední hodnotou, pokud se bude jednat o třídu

$$\mathbb{V} = \{ f(x; v) \},$$

pro kterou

$$f(x; v) = e^{x \cdot \theta(v) - \xi(\theta(v))} \cdot h(x),$$

kde $\mathbb{E}_v(\boldsymbol{X}) = \xi'(\theta(v)) = v$.

Budeme-li uvažovat vícerozměrnou NEF parametrizovanou střední hodnotou, pak jednotlivé složky musí být nezávislé a patřící do NEF parametrizované střední hodnotou.

Poznámka 3.23. Mezi důležitá rozdělení, která tvoří NEF parametrizovanou střední hodnotou, patří například normální rozdělení, exponenciální rozdělení a Poissonovo rozdělení. Jejich parametrické zapsání lze nalézt v tabulce 3.2.

Rozdělení pravděpodobnosti	$h(x)$	$\theta(x)$	$\xi(x)$
normální rozdělení	$\frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}$	$\frac{x}{\sigma^2}$	$\frac{x^2 \cdot \sigma^2}{2}$
exponenciální rozdělení	1	$-\frac{1}{x}$	$-\ln x$
Poissonovo rozdělení	$\frac{1}{x!}$	$\ln x$	e^x

Tabulka 3.2: Důležité NEF parametrizované střední hodnotou

Věta 3.24. *Mějme CE úlohu, kde* $S(\boldsymbol{x}) \geq 0$ *a nominální hustota pravděpodobnosti* $f(\boldsymbol{x};\boldsymbol{u})$ *patří do vícerozměrné NEF parametrizované střední hodnotou* $(\mathbb{V} = \{f(\cdot;\boldsymbol{v}) : \boldsymbol{v} \in \Theta \subset \mathbb{R}^n\})$*, přičemž optimální IS hustotu pravděpodobnosti hledáme ve* $\mathbb{V}$*. Pak řešení CE úlohy lze rozdělit na* n *podproblémů*

$$(\boldsymbol{v}^*)_j = \frac{\mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}) \cdot X_j)}{\mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}))}.$$

A řešení SC CE úlohy můžeme zapsat jako

$$\left(\widetilde{\boldsymbol{v}^*}\right)_j = \frac{\sum_{i=1}^N S(\boldsymbol{X}_i) \cdot W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{w}) \cdot X_{ij}}{\sum_{i=1}^N S(\boldsymbol{X}_i) \cdot W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{w})}.$$

Důkaz. Budeme-li předpokládat, že funkce

$$\phi(\boldsymbol{v}) = \mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot \ln f(\boldsymbol{X};\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}))$$

je konkávní a má pouze jedno globální maximum v bodě $\boldsymbol{v}^*$ (z jednoznačnosti výsledného řešení se ukáže, že je tento předpoklad oprávněný), můžeme zapsat CE úlohu jako

$$\boldsymbol{v}^* \text{ je řešením: } \mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot \nabla_{\boldsymbol{v}} \ln f(\boldsymbol{X};\boldsymbol{v}) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})) = 0. \tag{3.7}$$

Jelikož máme NEF parametrizovánu střední hodnotou (dle definice 3.22 jsou jednotlivé složky nezávislé), můžeme rozepsat sdruženou hustotu pravděpodobnosti $f(\boldsymbol{X};\boldsymbol{v})$ jako

$$f(\boldsymbol{X};\boldsymbol{v}) = \prod_{i=1}^n e^{X_i \cdot \theta_i(v_i) - \xi_i(\theta_i(v_i))} \cdot h_i(X_i).$$

Potom lze provést následující úpravy:

$$\begin{aligned}
\nabla_{\boldsymbol{v}} \ln f(\boldsymbol{X};\boldsymbol{v}) &= \nabla_{\boldsymbol{v}} \left(\sum_{i=1}^n X_i \cdot \theta_i(v_i) - \xi_i(\theta_i(v_i)) + \ln h_i(X_i) \right) \\
&= \left(X_i \cdot \theta_i'(v_i) - \xi_i'(\theta_i(v_i)) \cdot \theta_i'(v_i) \right)_i \; i = 1 \ldots n \\
&= \left(\theta_i'(v_i) \cdot (X_i - v_i) \right)_i \; i = 1 \ldots n.
\end{aligned}$$

V případě použití odvozeného výrazu ve formulaci CE úlohy (3.7) získáme (v netriviální případech je $\theta_i'(v_i) \neq 0$)

$$\begin{gathered}
\boldsymbol{v}^* = (v_i)_{i=1\ldots n} \text{ je řešením soustavy rovnic:} \\
\mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot \theta_i'(v_i) \cdot (X_i - v_i) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})) = 0, \; i = 1 \ldots n \\
\shortparallel \\
\mathbb{E}_{\boldsymbol{w}}(S(\boldsymbol{X}) \cdot (X_i - v_i) \cdot W(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w})) = 0, \; i = 1 \ldots n
\end{gathered}$$

Což je nejen soustava lineárních rovnic, ale navíc se v každé rovnici vyskytuje pouze jedna složka v_i. Proto lze vyjádřit explicitní vztah pro výpočet v_i

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot\left(X_i - v_i\right)\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right) &= 0 \\
\mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot v_i\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right) &= \mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot X_i\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right) \\
v_i\cdot\mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right) &= \mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot X_i\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right) \\
v_i &= \frac{\mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot X_i\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right)}{\mathbb{E}_{\boldsymbol{w}}\left(S\left(\boldsymbol{X}\right)\cdot W\left(\boldsymbol{X};\boldsymbol{u},\boldsymbol{w}\right)\right)}.
\end{aligned}$$

Čímž jsme dokázali tvrzení. □

Předchozí poznatky lze shrnout do následujícího algoritmu.

Algoritmus 3.25 (CE algoritmus pro NEF parametrizované střední hodnotou)**.**

Požadavky: $S\left(\boldsymbol{x}\right)\geq 0$ a $\mathbb{V}=\left\{f\left(\cdot;\boldsymbol{v}\right):\boldsymbol{v}\in\Theta\right\}$ je NEF parametrizovaná střední hodnotou

1. Volba počátečního $\widetilde{\boldsymbol{v}}_0=\boldsymbol{u}$, $t=1$.
2. Vygenerování náhodného výběru $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N\sim f\left(\cdot;\widetilde{\boldsymbol{v}}_{t-1}\right)$.
3. Aktualizace parametru
$$\widetilde{v}_{t,j}=\frac{\sum_{i=1}^{N}S\left(\boldsymbol{X}_i\right)\cdot W\left(\boldsymbol{X}_i;\boldsymbol{u},\widetilde{\boldsymbol{v}}_{t-1}\right)\cdot X_{ij}}{\sum_{i=1}^{N}S\left(\boldsymbol{X}_i\right)\cdot W\left(\boldsymbol{X}_i;\boldsymbol{u},\widetilde{\boldsymbol{v}}_{t-1}\right)}.$$
4. Pokud je splněna ukončovací podmínka (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1}\approx\widetilde{\boldsymbol{v}}_t$, ...), ukončit, jinak zpět ke kroku 2.

Je tedy zřejmé, že v případě NEF parametrizované střední hodnotou je CE metoda lepší volbou než VM metoda. Jednoduchost aktualizace parametrů umožňuje aplikaci CE metody na rozsáhlé problémy s velkou dimenzí vektoru parametrů.

Podobné zjednodušení CE metody lze provést pro diskrétní rozdělení pravděpodobnosti s konečným počtem prvků pravděpodobnostního prostoru $\mathcal{X}$.

Věta 3.26. *Mějme diskrétní rozdělení pravděpodobnosti s konečným počtem prvků pravděpodobnostního prostoru* $\mathcal{X}$

$$f\left(x;\boldsymbol{u}\right)=\sum_{j=1}^{m}u_j\cdot I\left(x=a_j\right),$$

kde u_j *jsou jednotlivé pravděpodobnosti výskytu hodnoty* $a_j\in\mathcal{X}$*. Dále mějme funkci* $S\left(x\right)\geq 0$ *a CE úlohu na hledání optimálního parametru*

$$\boldsymbol{v}\in\Theta=\left\{v\in\mathbb{R}^m:v_i\geq 0\wedge\sum_{i=1}^{m}v_i=1\right\}.$$

Pak řešení CE úlohy lze explicitně zapsat jako

$$v_j^* = \frac{\mathbb{E}_{\boldsymbol{w}} \left(S\left(\boldsymbol{X}\right) \cdot W\left(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}\right) \cdot I\left(\boldsymbol{X} = a_j\right) \right)}{\mathbb{E}_{\boldsymbol{w}} \left(S\left(\boldsymbol{X}\right) \cdot W\left(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}\right) \right)}$$

a příslušný SC CE úlohy jako

$$\widetilde{v}_j^* = \frac{\sum_{i=1}^N S\left(X_i\right) \cdot W\left(X_i; \boldsymbol{u}, \boldsymbol{w}\right) \cdot I\left(X_i = a_j\right)}{\sum_{i=1}^N S\left(X_i\right) \cdot W\left(X_i; \boldsymbol{u}, \boldsymbol{w}\right)}.$$

Důkaz. Jelikož v tomto případě provádíme optimalizaci přes veškerá pravděpodobnostní rozdělení definovaná na $\mathcal{X}$, bude řešení CE úlohy shodné s hustotou pravděpodobnosti s minimálním rozptylem z věty 3.7. Tedy hustota pravděpodobnosti nalezená CE metodou bude

$$\begin{aligned} f\left(x; \boldsymbol{v}^*\right) &= \frac{S\left(x\right) \cdot f\left(x; \boldsymbol{u}\right)}{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \right)} = \frac{\sum_{j=1}^m S\left(a_j\right) \cdot u_j \cdot I\left(x = a_j\right)}{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \right)} \\ &= \sum_{j=1}^m \left(\frac{S\left(a_j\right) \cdot u_j}{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \right)} \right) \cdot I\left(x = a_j\right) = \sum_{j=1}^m \left(\frac{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \cdot I\left(\boldsymbol{X} = a_j\right) \right)}{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \right)} \right) I\left(x = a_j\right) \end{aligned}$$

Z čehož vyplývá explicitní vztah pro parametry v_j^*

$$v_j^* = \frac{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \cdot I\left(\boldsymbol{X} = a_j\right) \right)}{\mathbb{E}_{\boldsymbol{u}} \left(S\left(\boldsymbol{X}\right) \right)} = \frac{\mathbb{E}_{\boldsymbol{w}} \left(S\left(\boldsymbol{X}\right) \cdot W\left(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}\right) \cdot I\left(\boldsymbol{X} = a_j\right) \right)}{\mathbb{E}_{\boldsymbol{w}} \left(S\left(\boldsymbol{X}\right) \cdot W\left(\boldsymbol{X}; \boldsymbol{u}, \boldsymbol{w}\right) \right)}.$$

Čímž jsme dokázali tvrzení. □

Poznámka 3.27. Jak již bylo zmíněno v důkazu předchozí věty, v případě diskrétní rozdělení pravděpodobnosti s konečným počtem prvků pravděpodobnostního prostoru $\mathcal{X}$ se řešení CE metody a VM metody kryjí a dosahují optimální hustoty pravděpodobnosti.

Poznámka 3.28. V případě vícerozměrného rozdělení pravděpodobnosti, kde jsou jednotlivé složky nezávislé diskrétní rozdělení pravděpodobnosti s konečným počtem prvků pravděpodobnostního prostoru $\mathcal{X}$, se CE úloha opět rozpadne na jednotlivé úlohy řešitelné dle věty 3.26.

3.3.3 Stabilizace konvergence CE metody - vyhlazovací parametr

Zabývejme se nyní konvergencí algoritmu 3.19. V případě, že používáme relativně malý počet pokusů v jednotlivých krocích algoritmu (N), můžeme narazit na kroky algoritmu, kdy se nově nalezená aproximace bude značně lišit. Takovéto skoky způsobí nízkou přesnost aproximace CE řešení a je nutné využít většího počtu pokusů.

Alternativní přístup, který zlepšuje konvergenci metody bez zvyšování výpočetní náročnosti, je metoda vyhlazování parametru. Princip metody je velice jednoduchý, místo

původní aproximace CE řešení jako

$$\widetilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(|S\left(\boldsymbol{X}_i\right)| \cdot \ln f\left(\boldsymbol{X}_i; \boldsymbol{v}\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right)\right)\right) \tag{3.8}$$

volíme novou aproximaci CE řešení jako lineární kombinaci řešení aktuální SC úlohy a předchozí aproximace CE řešení

$$\widetilde{\boldsymbol{v}}_t = \widetilde{\boldsymbol{v}}_{t-1} \cdot (1 - \alpha_t) + \alpha_t \cdot \overline{\boldsymbol{v}_t}, \tag{3.9}$$

kde $\alpha_t \in \langle 0, 1\rangle$ je vyhlazovací parametr pro vyhlazování parametru rozdělení $\boldsymbol{v}$ a $\overline{\boldsymbol{v}_t}$ spočteme dle vzorce 3.8.

Je zřejmé, že při volbě $\alpha_t = 1$ se řešení nezmění. Jednotlivé α_t se můžou lišit v každém kroku algoritmu (pro jednoduchost však budeme dále používat pouze konstantní α).

Algoritmus 3.29 (CE algoritmus s vyhlazování parametrů)**.**

1. Volba počátečního $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}$, $t = 1$ a posloupnosti $\{\alpha_t\}$.
2. Vygenerování náhodného výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim f\left(\cdot; \widetilde{\boldsymbol{v}}_{t-1}\right)$.
3. Aktualizace parametru dle 3.9.
4. Pokud je splněna ukončovací podmínka (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1} \approx \widetilde{\boldsymbol{v}}_t$, ...), ukončit, jinak zpět ke kroku 2.

Numerické testování vlivu volby vyhlazovacího parametru na přesnost CE řešení lze nalézt v příloze A. Tyto testy potvrzují zvyšující se přesnost aproximace při použití menšího vyhlazovacího parametru, avšak také nutnost použití vyššího počtu iterací k dosažení dané přesnosti.

Možností, jak snížit počet iterací, avšak zachovat přesnost, je použít místo konstantní α posloupnost danou předpisem

$$\alpha_t = \beta - \beta \cdot \left(1 - \frac{1}{t}\right)^q, \tag{3.10}$$

kde $\beta \in (0, 1)$ a $q \in \mathbb{R}^+$ jsou parametry.[1]

Využití metody vyhlazovacího parametru může tedy přinést vyšší stabilitu algoritmu. Avšak pro danou přesnost odhadu může být náročné nalézt správný poměr α a N k dosažení minimální výpočetní náročnosti algoritmu. Využití této metody je však nezbytné při aplikaci CE na optimalizační úlohy, neboť bez vyhlazovacího parametru může algoritmus dojít k špatnému výsledku (více v kapitole 5).

3.4 Problémy při použití Importance sampling - degenerace pravděpodobnosti

Tato část čerpá z [5]. V případě užití IS se můžeme často setkat s výskytem tzv. degenerace pravděpodobnosti. Jedná se o jev, kdy důležitý snímek (například s vysokou hodnotou $S(x)$) nabývá vysoké hodnoty LR.

Toto je způsobeno nevhodnou volbou IS, která sníží pravděpodobnost výskytu nějaké množiny důležitých snímku. Degenerace pravděpodobnosti se často objevuje v modelech, které mají mnoho optimalizovaných parametrů. V těchto případech i malá změna, avšak u velkého množství parametrů, může velmi ovlivnit výsledné LR.

Pro omezení výskytu degenerace pravděpodobnosti lze použít tzv. Screening metodu, která označí důležité (*„bottleneck"*) komponenty, jejichž parametry rozdělení má smysl měnit. [5]

Algoritmus 3.30 (Screening metoda)**.**

Inicializujme množinu důležitých komponent $B_0 = \{1, \ldots, n\}$ a jejich parametry rozdělení $\boldsymbol{u} = (u_1, \ldots, u_n)$. Vstupní parametry δ, d. Nastavme $t = 1$.

1. Vygenerování náhodného výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim f(\cdot; \boldsymbol{u})$.
2. Aktualizace parametru $\widetilde{\boldsymbol{v}}_t$ dle CE metody z definice 3.18.
3. Spočtení relativních změn
$$\delta_{t,i} = \frac{\widetilde{v}_{t,i} - u_i}{u_i}.$$
4. Sestavit aktualizaci množiny B_t jako množinu komponent z B_{t-1}, které splňují $\delta_{t,i} \geq \delta$. Pokud je $t \geq d$ ukončit, jinak $t = t + 1$ a zpět ke kroku 1.

---

4 Cross-entropy metoda pro RE úlohy

V této sekci si ukážeme použití CE metody na speciální případ sledované funkce a to $S(\boldsymbol{X}) = I(H(\boldsymbol{X}) \geq \gamma)$. Budou nás zajímat tzv. *„rare events"* (RE) úlohy, tedy takové, kde pozorování přinášející informace ($S(\boldsymbol{X}) = 1$) nastanou pouze s velmi nízkou pravděpodobností. Na začátek uveďme krátkou motivaci, proč je při řešení těchto RE úloh nutné využít jiný než klasický CMC přístup.

4.1 Motivace

Odhad chyby v CMC estimátoru vychází z centrální limitní věty. Chceme-li tedy spočítat počet potřebných pokusů pro požadovanou relativní přesnost, získáme vzorec:

$$RSO = \frac{z_{1-\alpha/2}\sqrt{s^2}}{\bar{X}\sqrt{N}} \rightarrow N \approx \frac{s^2}{RSO^2 \cdot \bar{X}^2},$$

kde s^2 je výběrový rozptyl, $z_{1-\alpha/2}$ je kvantil normálního rozdělení a $\bar{X}$ je výběrový průměr. Odhadujeme-li střední hodnotu $\boldsymbol{Y} = S(\boldsymbol{X}) = I(H(\boldsymbol{X}) \geq \gamma)$, jedná se o pravděpodobnost události $\mathbb{P}(H(\boldsymbol{X}) \geq \gamma)$. Jelikož je množina přípustných hodnot $\boldsymbol{Y} \in \{0, 1\}$, pro velká N tedy dostaneme:

$$N \approx \frac{Var(\boldsymbol{Y})}{RSO^2 \cdot \mathbb{E}(\boldsymbol{Y})^2} = \frac{\mathbb{E}(\boldsymbol{Y}^2) - \mathbb{E}(\boldsymbol{Y})^2}{RSO^2 \cdot \mathbb{E}(\boldsymbol{Y})^2} = \frac{1 - \mathbb{E}(\boldsymbol{Y})}{RSO^2 \cdot \mathbb{E}(\boldsymbol{Y})} \approx \frac{1}{RSO^2 \cdot \mathbb{E}(\boldsymbol{Y})},$$

neboť platí: $\mathbb{E}(\boldsymbol{Y}^2) = \mathbb{E}(\boldsymbol{Y}), \mathbb{E}(\boldsymbol{Y}) \sim 0$. Z tohoto odhadu získáme pro události s nízkou pravděpodobností výskytu $\mathbb{E}(\boldsymbol{Y}) = \mathbb{P}(H(\boldsymbol{X}) \geq \gamma) \sim 10^{-6}$ a rozumnou relativní odchylkou $RSO = 0.01$ nutnost provést obrovské množství pokusů:

$$N \approx 10^{10}.$$

Což je výpočetně neproveditelné, proto je třeba využít jiného přístupu než CMC.

4.2 Obecný přístup použití CE pro Rare-Event systémy

V předchozí kapitole jsme ukázali obecný přístup, jak pomocí CE metody nalézt „optimální" IS hustotu pravděpodobnosti pro estimaci $\mathbb{E}(S(\boldsymbol{X}))$ z parametrické pravděpodobnostní rodiny $\mathbb{V} = \{f(\cdot; \boldsymbol{v}) : \boldsymbol{v} \in \Theta\}$. Postupně jsme odvodili vztah pro aproximaci pomocí SC přístupu

$$\widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v} \in \Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(|S(\boldsymbol{X}_i)| \cdot \ln f(\boldsymbol{X}_i; \boldsymbol{v}) \cdot W(\boldsymbol{X}_i; \boldsymbol{u}, \boldsymbol{w}) \right) \right).$$

V případě tvaru $S(\boldsymbol{X}) = I(H(\boldsymbol{X}) \geq \gamma)$ dostaneme

$$\widetilde{\boldsymbol{v}^*} = \underset{\boldsymbol{v} \in \Theta}{\operatorname{argmax}} \left(\frac{1}{N} \cdot \sum_{i=1}^{N} \left(I(H(\boldsymbol{X}_i) \geq \gamma) \cdot \ln f(\boldsymbol{X}_i; \boldsymbol{v}) \cdot W(\boldsymbol{X}_i; \boldsymbol{u}, \boldsymbol{w}) \right) \right),$$

z čehož je zřejmé, že v případě RE, kde je pravděpodobnost $\mathbb{P}\left(I\left(H\left(\boldsymbol{X}\right)\geq\gamma\right)=1\right)$ velmi malá, tento SC přístup zhavaruje. Toto je dáno tím, že pro malé počty pokusů N a nízkou pravděpodobnost $\mathbb{P}\left(I\left(H\left(\boldsymbol{X}\right)\geq\gamma\right)=1\right)$ bude velmi často

$$\frac{1}{N}\cdot\sum_{i=1}^{N}\left(I\left(H\left(\boldsymbol{X}_i\right)\geq\gamma\right)\cdot\ln f\left(\boldsymbol{X}_i;\boldsymbol{v}\right)\cdot W\left(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{w}\right)\right)=0.$$

Pro vyřešení tohoto problému je navržená tzv. dvou kroková CE metoda. Tato metoda je principiálně velmi jednoduchá, jedná se o změnu parametru γ tak, aby $I\left(H\left(\boldsymbol{X}\right)\geq\gamma\right)$ nabývalo častěji hodnoty 1. A tedy příslušný SC přinášel nové informace o aproximované hodnotě. Nový parametr γ musí být volen s ohledem právě na počet nenulových hodnot $I\left(H\left(\boldsymbol{X}\right)\geq\gamma\right)$.

Definice 4.1 (Adaptivní volba parametru γ)**.** Mějme SC CE metody pro RE úlohu, kde $S\left(\boldsymbol{X}\right)=I\left(H\left(\boldsymbol{X}\right)\geq\gamma\right)$, daný počet pokusů N a požadovaný počet hodnot pro smysluplnost SC aproximace $\rho\cdot N$, kde $\rho\in\left(0,1\right)$. Volbu parametru $\tilde{\gamma}$ můžeme provést jako $1-\rho$ kvantil hodnot $H_i=H\left(\boldsymbol{X}_i\right)$

$$\tilde{\gamma}=H_{(\lceil(1-\rho)\cdot N\rceil)}.$$

Je zřejmé, že jedno řešení takovéto upravené SC nebude dobře aproximovat optimální CE parametr IS hustoty pravděpodobnosti. Proto opět využijeme iteračního přístupu. Shrňme metodu do následujícího algoritmu.

Algoritmus 4.2 (Obecný postup CE pro RE)**.**

1. Inicializace: počáteční hodnoty $\widetilde{\boldsymbol{v}}_0=\boldsymbol{u}$, $t=1$, ρ, N.
2. Vygenerování náhodných pokusů $\boldsymbol{X}_1,\boldsymbol{X}_2,...,\boldsymbol{X}_N\sim f\left(\cdot;\widetilde{\boldsymbol{v}}_{t-1}\right)$.
3. Spočtení $H_i=H\left(\boldsymbol{X}_i\right)$, jejich seřazení a zjištění $(1-\rho)$ kvantilu pro spočtení $\tilde{\gamma}_t=\min\left(H_{\lceil(1-\rho)\cdot N\rceil},\gamma\right)$.
4. Spočtení $\widetilde{\boldsymbol{v}}_t$ vyřešením:

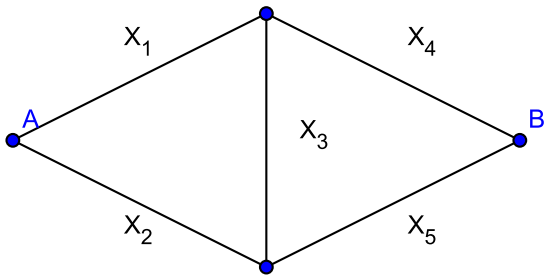
$$\widetilde{\boldsymbol{v}}_t=\underset{\boldsymbol{v}\in\Theta}{\operatorname{argmax}}\left(\frac{1}{N}\sum_{i=1}^{N}I\left(H\left(\mathbf{X}_i\right)\geq\tilde{\gamma}_t\right)W\left(\mathbf{X}_i;\boldsymbol{u},\widetilde{\boldsymbol{v}}_{t-1}\right)\ln f\left(\mathbf{X}_i;\boldsymbol{v}\right)\right).$$

5. Pokud $\tilde{\gamma}_t=\gamma$ a zároveň je splněna ukončovací podmínka (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1}\approx\widetilde{\boldsymbol{v}}_t$,...) konec algoritmu, jinak se vracíme ke kroku 2.

Poznámka 4.3. Volba ρ záleží na charakteru problému. Příliš velké hodnoty ρ můžou způsobit stagnaci algoritmu (používané hodnoty $\tilde{\gamma}_t$ budou stále $\tilde{\gamma}_t \ll \gamma$), příliš malé hodnoty (stále však za předpokladu $\rho \cdot N \geq 1$) mohou způsobit nestabilitu aproximace a vytvořit IS trpící degenerací pravděpodobnosti.

Uveďme aplikaci algoritmu 4.2 na ukázkovém příkladě.

Příklad 4.4. Mějme silniční síť danou následujícím obrázkem.



Obrázek 4.1: Znázornění cest (hledáme nejkratší cestu z A do B)

Dále uvažujme, že časy na překonání úseků X_1, X_2, X_3, X_4, X_5 jsou dány exponenciálními rozděleními se středními hodnotami $\boldsymbol{u} = (u_1, ..., u_5)$. Tedy sdružená hustota pravděpodobnosti rozdělení časů na překonání jednotlivých cest je dána:

$$f(\boldsymbol{x}; \boldsymbol{u}) = \exp\left(-\sum_{i=1}^{5} \frac{x_i}{u_i}\right) \prod_{i=1}^{5} \frac{1}{u_i}.$$

Úkolem je zjistit pravděpodobnost, že nejkratší trasa z A do B bude trvat déle než zadaná hodnota γ. Je vidět, že bude-li hodnota γ dosti vysoká, bude hledaná pravděpodobnost velmi nízká, jedná se tedy o RE úlohu.

Pro vyřešení úlohy použijeme nejprve přístup CMC, poté CE a oba přístupy srovnáme. Úlohu budeme řešit pro $\boldsymbol{u} = (0.25, 0.4, 0.1, 0.3, 0.2)$ a $\gamma = 2$.

Výsledky CMC pro jednotlivé počty pokusů lze vidět v tabulce 4.1.

Počet NP	10^5	10^6	10^7	10^8
Výběrový průměr	1.0000e-05	1.7000e-05	1.4200e-05	1.3560e-05
Relativní odchylka (95%)	1.6449	0.3989	0.1380	0.0447
Čas výpočtu	0.033231	0.297059	3.002615	29.457663

Tabulka 4.1: Výsledky ukázkové úlohy RE pro CMC

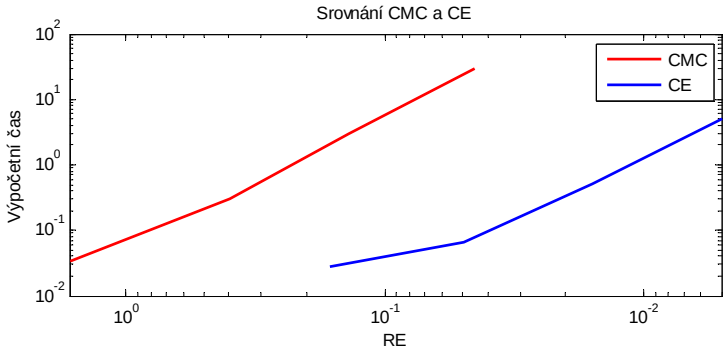
Při použití CE metody použijeme algortimus 4.2 kombinovaný s poznatky z věty 3.24, neboť se jedná o příklad NEF parametrizované střední hodnotou. Výsledky IS s použitím CE metody lze vidět v tabulce 4.2.

Počet NP	10^4	10^5	10^6	10^7
Kroků CE	5	4	4	4
Výběrový průměr	1.2725e-05	1.3314e-05	1.3534e-05	1.3420e-05
Relativní odchylka (95%)	0.1621	0.0494	0.0158	0.0050
Čas výpočtu	0.028295	0.067131	0.510228	5.099259

Tabulka 4.2: Výsledky ukázkové úlohy RE pro CE

Jak lze vidět z výsledků CMC, pro rozumný odhad pravděpodobnosti, je třeba vysokého počtu pokusů. Což je zřejmé z velice nízké pravděpodobností události $\sim 10^{-5}$, tedy máme-li méně než 10^5 pokusů, často nejsme schopni pravděpodobnost ani hrubě odhadnout (náhodné pokusy jsou samé 0).

V případě výsledků testů CE metody lze pozorovat značné zlepšení odhadu. Pro názorné srovnání uveďme ještě graf závislosti výpočetního času na výsledné relativní přesnosti řešení obou metod (jedná se o spravedlivější porovnání těchto metod, neboť v CE metodě je započítán i čas strávený při řešení samotné CE úlohy).



Obrázek 4.2: Srovnání CMC a CE pro RE systém

Jak lze vidět v grafu srovnání metod, je CE pro vyšší přesnosti přibližně 400krát efektivnější. △

4.3 Adaptivní CE metoda

V případě algoritmu 4.2 je nutnost správné volby parametrů N a ρ. V případě, že tyto parametry nevolíme správně (zpravidla nízké N) nemusí CE algoritmus dojít k rozumnému odhadu „optimálních" parametrů.

Pro odstranění této nevýhody je navržena adaptivní verze algoritmu 4.2, kde parametry ρ a N postupně měníme v běhu programu. Jedná se konkrétně o tyto dvě úpravy:

1. Snižování ρ pokud není dosaženo dostatečného růstu $\tilde{\gamma}$.

2. Zvyšování počtu pokusů N, pokud je již ρ příliš nízké (příliš málo hodnot $\lfloor \rho \cdot N \rfloor$).

Algoritmus 4.5 (Adaptivní CE algoritmus pro RE úlohy)**.**

1. Definujme $\gamma, \delta, \beta, M, \widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}, \rho_0 = \rho, N_0 = N$ a $t = 1$.
2. Vygenerování náhodných pokusů $X_1, X_2, ..., X_{N_0} \sim f(\boldsymbol{X}; \widetilde{\boldsymbol{v}}_{t-1})$, spočtení výkonosti $H_i = H(X_i)$, jejich seřazení a spočtení $\tilde{\gamma}_t = \min\left\{H_{(\lceil(1-\rho_{t-1})\cdot N\rceil)}, \gamma\right\}$.
3. Spočteme aktualizaci $\widetilde{\boldsymbol{v}}_t$:
$$\widetilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}}{\operatorname{argmax}} \left(\frac{1}{N_{t-1}} \sum_{i=1}^{N_{t-1}} I\left(H(X_i) \geq \tilde{\gamma}_t\right) \cdot W\left(X_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right) \cdot \ln f\left(X_i; \boldsymbol{v}\right) \right).$$
4. Pokud je $\tilde{\gamma}_t = \gamma$ a jsou splněny ukončovací podmínky (pevný počet kroků, $\widetilde{\boldsymbol{v}}_{t-1} \approx \widetilde{\boldsymbol{v}}_t$,...), pak ukončit program, jinak pokračovat.
5. Zjistit, zdali existuje $\hat{\rho}$ takové, že $(1-\hat{\rho})$ kvantil H_i je větší nebo roven $\min\{\gamma, \tilde{\gamma}_t + \delta\}$ a zároveň je $\hat{\rho} \cdot N_{t-1} \geq M$. Pokud ano ρ_t nastavíme jako největší z takovýchto $\hat{\rho}$ a $N_t = N_{t-1}$, jinak $N_t = \beta \cdot N_{t-1}$. Dále pokračujeme krokem 2.

Poznámka 4.6 (Parametry algoritmu 4.5). Pro lepší orientaci v algoritmu 4.5 uveďme popis jednotlivých volitelných parametrů algoritmu.

N - počáteční počet pokusů v jednotlivých krocích, vhodná hodnota může být například $5 \cdot n$, kde n je velikost vektoru $\boldsymbol{u}$

δ - minimální požadovaný nárůst parametru $\tilde{\gamma}$ v jednotlivých krocích algoritmu, vhodná hodnota může být například γ/c, kde c je např. $\{10, 20, 30, ...\}$

β - poměr zvyšování počtu pokusů v případě nutnosti $N_t = \beta \cdot N_{t-1}$,vhodná volba může být $\beta \sim 1.5$, lze použít alternativní formule $N_t = \beta + N_{t-1}$

M - minimální počet hodnot $I(H(X_i) \geq \tilde{\gamma}) = 1$, ve většině případů bude stačit $M = 1$, $\hat{\rho} \cdot N_{t-1} \geq M$ lze nahradit alternativní formulí $\hat{\rho} \geq \xi$

Dále si ukážeme aplikaci předchozích poznatků na rozsáhlé RE úloze z praxe.

4.4 Použití CE metody pro urychlení rozsáhlých simulací

Jelikož jednou z hlavních aplikací CE metody je urychlení rozsáhlých simulací, je vhodné uvést i takovýto příklad. Zároveň se ukáže, jak si CE metoda vede na reálných úlohách a jaké výhody získáme jejím použitím a případnou kombinací s masivní GPU paralelizací.

Zkoumanou úlohou bude určení velikosti finanční rezervy, kterou potřebuje banka pro pokrytí případných ztrát způsobených insolvencí dlužníků. Nejprve je dobré uvést základní pojmy v této problematice.

- *„Credit risk"* (CR) - neboli úvěrové riziko, značí hrozbu krachu společnosti na základě neplnění finančních závazků (insolvence) jejich protistran (dlužníků),
- *„Economic capital"* (EC) - značí finanční rezervu chránící společnost proti nečekaným ztrátám během daného časového intervalu.
- *„Credit risk economic capital"* (CREC) - je velikost finanční rezervy pro pokrytí ztrát způsobených:
  - krachem protistrany,
  - snížením úvěruschopnosti protistrany, nebo
  - nemožností protistrany plnit finanční závazky z důvodu krachu (insolvence) státu, ve kterém působí.

Jelikož se jedná o aplikaci tohoto modelu v bankovní sféře, je vyžadována vysoká spolehlivost. Cílem modelu je určit velikost finanční rezervy tak, aby možnost krachu nastala maximálně s 0.005% pravděpodobností.

Detailní popis modelu přesahuje rámec této práce a proto je umístěn v příloze, viz sekce B. Zde uvedeme pouze hrubý popis modelu dostačující pro studium aplikace CE metody.

4.4.1 Popis modelu

CREC model je modifikací více-faktorového Mertonova modelu (viz [18]) a zakládá se na simulaci ekonomických scénářů jakožto vícerozměrného Markovova řetězce. Jednotlivé Markovovy řetězce reprezentují vývoj jednotlivých závazků v čase, přičemž stavy označují známku (*„rating"*), která nese informaci o stavu daného závazku. Sledovaným stavem je stav krachu, který je absorpční, a způsobuje finanční ztrátu. Tato ztráta není fixní, ale je simulována jako náhodná část závazku (finanční obnos) v času krachu.

Jednotlivé Markovovy řetězce simulující závazky jsou mezi sebou silně korelované, tato korelace je řešena simulačně.

Jelikož data simulace (korelační závislosti, matice přechodu) jsou v praxi závislé na čase a tedy nedostupné pro další kroky simulace, jsou řetězce simulovány pouze jedním krokem. Zbytek simulace je pouze aproximován podle zadaných formulí.

Pro další práci s tímto modelem stačí uvažovat ztrátovou funkci $H(\boldsymbol{X})$, která vyjadřuje ztrátu spočtenou modelem na základě vstupního vektoru $\boldsymbol{X} = (X_i)$, přičemž model využívá k simulaci pouze náhodná čísla z normovaného normálního rozdělení, tedy $X_i \sim \mathcal{N}(0, 1)$. Dimenze rozdělení je v případě zkoumaného portfólia banky ~ 7600.

4.4.2 Implementace

Nejprve byl model implementován v Matlabu (kódy jsou přiloženy na CD, příloha G). Celková simulace je rozdělena na dvě hlavní části:

1. Příprava dat k simulaci.

- matice přechodu jsou převedeny na Z-score matice (kvantily normálního rozdělení), toto odstraňuje nutnost aplikace distribuční funkce při běhu simulace;
- odhady ztrát, pokud se po prvním kroku řetězce nezmění známka země ani závazku;
- přeindexování polí s daty pro simulaci tak, aby index (na vstupu pořadové číslo v portfoliu banky) odpovídal pozici v poli;
- ....

2. Samotná simulace.

   - generování náhodných čísel;
   - simulace korelace mezi jednotlivými závazky;
   - spočtení kroku MŘ a odhad finančních ztrát.

Jelikož se jedná o složitou simulaci s požadavkem na vysoký počet pokusů (pro požadovanou přesnost je v praxi nutné 10^7 pokusů), je časová náročnost algoritmu velmi vysoká (v rámci hodin).

I přes využití paralelního cyklu ***„parfor"*** přes jednotlivé pokusy, které na 4 jádrovém procesoru urychlilo výsledek přibližně 3,5 krát a značné vektorizace byl čas výpočtu stále příliš vysoký (4.5 hodin). Což také byla prvotní motivace k pararelizaci na GPU a využití CE metody.

Informace o paralelní implementaci na GPU a srovnání výkonosti jsou uvedeny v samostatné části, viz sekce 8.2. Z důvodu velké časové náročnosti CPU implementace je pro veškeré další testování použita GPU implementace.

4.4.3 Aplikace CE metody

Jelikož se úloha oproti předchozím RE úlohám liší, nyní řešíme úlohu pro dané ℓ najdi γ takové, že platí:

$$\mathbb{P}\left(H\left(\boldsymbol{X}\right) \geq \gamma\right) = \mathbb{E}\left(I\left(H\left(\boldsymbol{X}\right) \geq \gamma\right)\right) = \ell,$$

kde ℓ je zadaná pravděpodobnost 0.005%, $H\left(\boldsymbol{X}\right)$ je celková ztráta spočtená simulací a γ je hledaný kvantil, je třeba použít jiný algoritmus. Algoritmus pro řešení této úlohy lze získat úpravou algoritmu 4.2. [1]

Algoritmus 4.7 (CE metoda pro hledání kořene)**.**

Mějme $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}$, $t = 1$, ρ, ℓ.

1. Vygenerujme náhodné pokusy $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N \sim f\left(\cdot, \widetilde{\boldsymbol{v}}_{t-1}\right)$.
2. Spočtení výkonosti $H_i = H\left(\boldsymbol{X}_i\right)$, jejich seřazení a zjištění $(1-\rho)$ kvantilu $\tilde{\gamma}_t = H_{\lceil (1-\rho)\cdot N \rceil}$.

3. Dále spočteme pravděpodobnost $S_i \geq \tilde{\gamma}_t$:

$$\ell_t = \frac{1}{N} \sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \tilde{\gamma}_t\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right).$$

4. Spočteme aktualizaci $\widetilde{\boldsymbol{v}}_t$:

$$\widetilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}}{\operatorname{argmax}} \left(\frac{1}{N} \sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \tilde{\gamma}_t\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right) \cdot \ln f\left(\boldsymbol{X}_i; \boldsymbol{v}\right) \right).$$

5. Pokud je $\ell_t \geq \ell$ ukončit program, jinak zpět do bodu 2.

V případě zadané úlohy veškerá náhodná čísla pocházejí z $\mathcal{N}(0,1)$. V předchozí části bylo odvozeno, že v případě NEF parametrizované střední hodnotou je formule pro řešení SC CE úlohy velmi jednoduchá (viz věta 3.24). Proto jako rodinu pravděpodobnostních rozdělení, ve které budeme hledat optimální IS, zvolíme vícerozměrné normální rozdělení s jednotkovým rozptylem parametrizované středními hodnotami

$$X_i \sim \mathcal{N}\left(v_i, 1\right).$$

Tedy vektor počátečních parametrů $\boldsymbol{u}$ je nulovým vektorem.

Jelikož se jedná o rozsáhlý problém, velikost vektoru optimalizovaných parametrů je přibližně 7600, tento základní algoritmus (algoritmus 4.7) by konvergoval jen velmi špatně a jím nalezené řešení by vykazovalo značné známky degenerace pravděpodobnosti. Proto je třeba využít dříve uvedené modifikace základní metody pro zlepšení konvergence a zabránění degenerace pravděpodobnosti, konkrétně použijeme následující modifikace:

- Vyhlazovací parametr α podle formule (3.10).
- Adaptivní verzi algoritmu pro RE (viz algoritmus 4.5).
- Modifikaci Screening metody (viz algoritmus 3.30), pro použití v průběhu CE metody.

Poslední zmíněnou modifikaci Screening metody použijeme jako úpravu počtu aktualizovaných parametrů:

Algoritmus 4.8 (Modifikace Screening metody pro použití při běhu CE metody)**.**

---

1. Mějme $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}, t = 1, \rho, \ell, a, b, \psi$.
2. Vygenerujme náhodné pokusy $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N \sim f\left(\cdot, \widetilde{\boldsymbol{v}}_{t-1}\right)$.

3. Spočteme aktualizaci $\widetilde{\boldsymbol{v}}_t$.
4. Ve vektoru $\widetilde{\boldsymbol{v}}_t$ necháme pouze hodnoty $|\widetilde{v}_{t,i} - u_i| \geq \psi$, maximálně však $(a + b \cdot t)$ hodnot s největší hodnotou $|\widetilde{v}_{t,i} - u_i|$. Zbytek nahradíme původními hodnotami u_i.
5. Ukončovací podmínka.

Tedy je stanovena minimální hranice změny parametru ψ a zároveň maximální počet parametrů s největší změnou oproti nominálním parametrům a. Tento počet se navíc může s dalšími iteracemi zvětšovat vzhledem k parametru b.

Spojením metody pro hledání kořene se všemi zmíněnými modifikacemi získáme následující metodu (v algoritmu již přepokládáme parametry řešené úlohy, tedy NEF parametrizovanou střední hodnotou $\boldsymbol{u} = \overline{\boldsymbol{o}}$).

Algoritmus 4.9 (Modifikovaná CE metoda pro CREC model)**.**

Na vstupu přichází zadání problému: $\ell, \widetilde{\boldsymbol{v}}_0 = \boldsymbol{u} = \overline{o}$ a volitelné parametry algoritmu

$$N_0, \rho_0, \beta \in (0,1), q > 0, \psi, M, \xi, a, b, \delta < 1, \varepsilon.$$

Inicializace $t = 1$.

1. Vygenerování náhodných pokusů $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_{N_t} \sim f(\cdot, \widetilde{\boldsymbol{v}}_{t-1}) = \mathcal{N}(\widetilde{\boldsymbol{v}}_{t-1}, 1)$.
   (a) Spočtení ztráty $H_i = H(\boldsymbol{X}_i)$.
   (b) Seřazení a zjištění $(1-\rho_t)$ kvantilu $\overline{\gamma_t} = H_{\lceil (1-\rho_t) \cdot N_t \rceil}$.
   (c) Přeuspořádání hodnot $\boldsymbol{X}_i$, podle velikosti hodnot $H(\boldsymbol{X}_i)$ sestupně, do posloponosti $\widetilde{\boldsymbol{X}_j}$.
   (d) Spočtení hodnot
   $$\widetilde{W_i} = \frac{1}{N} \sum_{j=1}^{i} W\left(\widetilde{\boldsymbol{X}_j}; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right).$$
   (e) Nalezení hodnoty k jako první index i, pro který platí $\widetilde{W_i} \geq \ell$.
   (f) Nastavení
   $$\tilde{\gamma}_t = \min\left\{\overline{\gamma_t}, H\left(\widetilde{\boldsymbol{X}_k}\right)\right\},$$
   $$\ell_t = \frac{1}{N} \sum_{i=1}^{N} I\left(H(\boldsymbol{X}_i) \geq \tilde{\gamma}_t\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right).$$
2. Spočtení vyhlazovacího parametru
   $$\alpha_t = \beta - \beta \cdot \left(1 - \frac{1}{t}\right)^q.$$

3. Spočtení aktualizace $\widetilde{\boldsymbol{v}}_t$:

$$\widetilde{v}_{t,j} = \widetilde{v}_{t-1,j} \cdot (1 - \alpha_t) + \alpha_t \cdot \left(\frac{\sum_{i=1}^{N_t} I\left(H\left(\boldsymbol{X}_i\right) \geq \tilde{\gamma}_t\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right) \cdot X_{ij}}{\sum_{i=1}^{N_t} I\left(H\left(\boldsymbol{X}_i\right) \geq \tilde{\gamma}_t\right) \cdot W\left(\boldsymbol{X}_i; \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{t-1}\right)} \right).$$

4. Aplikování restrikce počtu optimalizovaných parametrů: ponechání pouze hodnot $\widetilde{v}_{t,j} \geq \psi$, maximálně však $(a + b \cdot t)$ hodnot s největší hodnotou $\widetilde{v}_{t,j}$, vynulování ostatních.

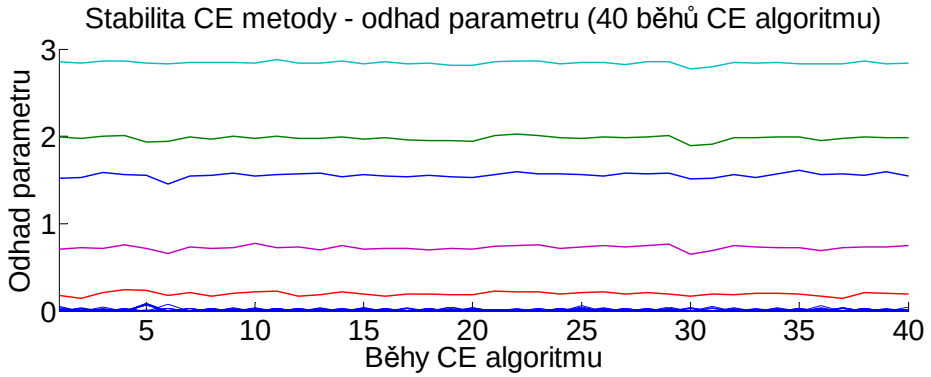
5. Pokud $\ell_t \geq \ell$ a zároveň $\frac{|\widetilde{\boldsymbol{v}}_t - \widetilde{\boldsymbol{v}}_{t-1}|}{|\widetilde{\boldsymbol{v}}_t|} \leq \varepsilon$ ukončit program, jinak pokračovat.

6. Zjistit, zdali existuje $\hat{\rho}$ takové, že $\hat{\rho}$ kvantil $\widetilde{W}_i$ je větší nebo roven $\max\{\ell, \ell_t \cdot \delta\}$ a zároveň je $\hat{\rho} \cdot N_{t-1} \geq M$. Pokud ano, ρ_t nastavíme jako největší z takovýchto $\hat{\rho}$ a $N_t = N_{t-1}$, jinak $N_t = \xi + N_{t-1}$. Dále pokračujeme krokem 1.

4.4.4 Numerické výsledky

Dále otestujeme navržený algoritmus 4.9. Zjistíme, zdali se algoritmem nalezené hodnoty neliší mezi jednotlivými běhy a otestujeme velikost redukce rozptylu s nalezeným IS rozdělením.

Stabilita navržené metody Nejprve ukažme, zdali je algoritmus stabilní. Toto bylo testováno opakovaným během algoritmu na stejných vstupních datech a porovnáním výsledků.



Obrázek 4.3: Stabilita CE algoritmu pro CREC model

Jak lze vidět na obrázku 4.3. nalezené velikosti středních hodnot jsou ve více bězích algoritmu téměř totožné. Toto kromě spolehlivosti algoritmu ukazuje, že nalezené parametry budou blízko skutečnému minimu. Také je vidět, že pouze 5 parametrů rozdělení má výrazný vliv na výslednou ztrátu, tyto parametry náleží hodnotám řídící korelace mezi jednotlivými závazky.

Parametry algoritmu, pro které byly testy prováděny, lze vidět v tabulce 4.3, doba běhu a počty iterací jsou uvedeny v tabulce 4.4.

N_0	ρ_0	β	q	ψ	M	ξ	a	b	δ	ε
4608	0.01	0.5	5	0.001	1	4608	1	0.5	0.95	0.01

Tabulka 4.3: Parametry CE algoritmu pro CREC model

Čas průměr	Čas maximum	Počet iterací průměr	Počet iterací maximum
10.7 s	14.6 s	19.8	27

Tabulka 4.4: Doba běhu a počty iterací (pro 40 běhů algoritmu)

Je nutno uvést, že čas běhu a počet iterací se bude při změně některých parametrů velice lišit. Na počet iterací mají především vliv parametry: ε, $\alpha_t = \beta - \beta \cdot \left(1 - \frac{1}{t}\right)^q$ a δ. Zvýšením ε a α_t nebo snížením δ lze CE metodu značně urychlit, zvyšuje se však možnost špatné konvergence algoritmu.

Redukce rozptylu Dále ukažme, jaké redukce rozptylu lze při použití nalezené IS hustoty pravděpodobnosti dosáhnout. Jelikož však odhad rozptylu kvantilu je v případě IS velice náročný (o této tématice pojednává [19]), použijeme pro výpočet více běhů algoritmu a hledaný rozptyl spočteme jako výběrový rozptyl spočtených hodnot γ.

Pro tento výpočet použijeme z důvodu časové náročnosti 10^6 pokusů a 100 běhů algoritmu. Dá se předpokládat, že rozptyl hledané hodnoty bude mít lineární závislost s počtem pokusů.

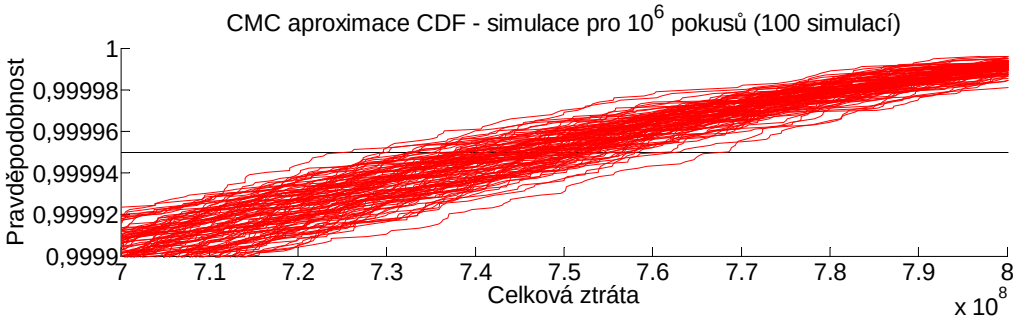
Jednotlivé rozptyly lze vidět v následující tabulce.

CMC	IS
$5.9980 \cdot 10^{13}$	$1.8676 \cdot 10^{11}$

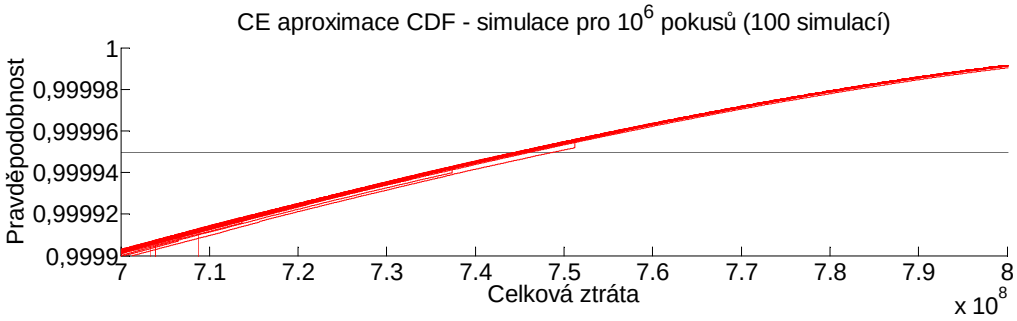
Tabulka 4.5: CREC model: rozptyl CMC a IS pro 10^6 pokusů (100 běhů algoritmu)

Jak lze vidět dosáhli jsme značného zpřesnění odhadu. Poměr rozptylů je 321, což znamená, že pro stejný počet pokusů lze získat přibližně 18 krát přesnější výsledek (založeno na směrodatné odchylce). V případě požadování stejné přesnosti stačí oproti CMC použít 321 krát méně pokusů.

Déle uveďme grafy jednotlivých simulovaných distribučních funkcí pro CMC a IS.



(a) Výsledky pro CMC



(b) Výsledky pro CE

Obrázek 4.4: Porovnání rozptylu odhadu kvantilu v CMC a CE pro 30 běhů algoritmu

V grafech lze názorně vidět získanou redukci rozptylu. V grafu pro IS lze také vidět lehké známky degenerace pravděpodobnosti, které se projevují jako vertikální skoky distribuční funkce. Tato degenerace pravděpodobnosti značně zhoršuje rozptyl hledané hodnoty.

Nabízí se tedy možnost odebrat náhodné pokusy vykazující příliš velké LR. Pokud toto provedeme (odebereme náhodné pokusy s LR 1000 krát větším, než je průměr LR) pro distribuční funkce získané předchozími simulacemi, získáme rozptyl $6.0181 \cdot 10^{10}$, což je 997 krát menší rozptyl, než u CMC estimátoru. Tímto jsme odstranili pouze 1 pokus z miliónu, je však nutné podotknout, že tento přístup vede k estimátoru, který již nebude nestranný.

Celkové urychlení simulace použitím kombinace CE + GPU akcelerace lze najít v sekci 8.2.

5 Cross-entropy metoda pro řešení kombinatorických optimalizací

V této části ukážeme, jak lze jednoduše modifikovat CE metodu pro RE na efektivní a univerzální nástroj pro řešení kombinatorických optimalizací. Většina těchto optimalizačních úloh, jak deterministických, tak stochastických, například: problém maximálního řezu (*„max-cut problem"*), problém obchodního cestujícího (*„the traveling salesman problem"* TSP), problém seřazení posloupností (*„sequence alignment problém"*), problém alokace bufferu (*„buffer allocation problem"*), jsou NP-kompletní (NP-complete) nebo NP-těžké (NP-hard)[4] problémy.

Dalšími dobře známými přístupy na řešení kombinatorických optimalizací jsou například simulované žíhání, nebo genetické algoritmy.

5.1 CE metoda jako obecný nástroj pro optimalizace

Předpokládejme, že chceme maximalizovat hodnoty výkonnostní funkce $H(\boldsymbol{x})$ přes množinu všech přípustných hodnot $\mathcal{X}$. Tedy hledáme maximální možný výkon funkce $H(\boldsymbol{x})$

$$\gamma^* = \max_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x})$$

společně s argumentem dosahující této hodnoty

$$\boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}).$$

Pro řešení takovéhoto problému musíme nejprve definovat přidružený stochastický problém (*„associated stochastic problem"* ASP).

Definice 5.1 (Přidružený stochastický problém (ASP))**.** Mějme optimalizační úlohu

$$\gamma^* = \max_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}).$$

Dále mějme parametrickou rodinu pravděpodobnostních rozdělení

$$\mathbb{V} = \{f(\cdot;\boldsymbol{v}), \boldsymbol{v}\in\Theta\}$$

definovanou na $\mathcal{X}$ a nějaký odhad řešení $\gamma \leq \gamma^*$. Přidruženým stochastickým problémem rozumíme úlohu

$$\ell(\gamma) = \mathbb{P}_{\boldsymbol{u}}(H(\boldsymbol{X}) \geq \gamma) = \mathbb{E}_{\boldsymbol{u}}(I(H(\boldsymbol{X} \geq \gamma))),$$

kde $\boldsymbol{X} \sim f(\cdot;\boldsymbol{u})$ je náhodný vektor (například z Bernoulliho rozdělení) a γ může být známý či neznámý parametr (v případě neznámého problému máme zadanou hodnotu $\ell(\gamma)$).

[4] NP-hard je problém na který se dají redukovat veškeré NP problémy (nedeterministické polynomiální problémy - jejich řešení se dá ověřit v polynomiálním čase), tedy problém složitý nejméně tak, jako nejtěžší NP problém. NP-complete je třída problémů patřící do NP-hard, které jsou zároveň stále NP problémy .

Pomocí tohoto ASP můžeme iteračně řešit zadanou maximalizační úlohu. To provedeme postupnou úpravou ASP pomocí postupně zvětšujícího se $\gamma_t \leq \gamma^*$:

$$\mathbb{E}_{\boldsymbol{u}} \left(I \left(H \left(\boldsymbol{X} \geq \gamma_t \right) \right) \right).$$

Tato posloupnost ASP však závisí na uměle zvoleném parametru $\boldsymbol{u}$, který primárně nesouvisí s optimalizačním problémem. Proto budeme volit posloupnost ASP definovanou jako

$$\mathbb{E}_{\boldsymbol{v}_{t-1}} \left(I \left(H \left(\boldsymbol{X} \geq \gamma_t \right) \right) \right),$$

kde $\boldsymbol{v}_{t-1}$ je vektor parametrů získaných CE metodou aplikovanou na ASP v $(t-1)$-ní iteraci. Tímto postupem získáme algoritmus nazývaný jako deterministická verze CE.

Algoritmus 5.2 (Deterministická CE metoda pro optimalizace)**.**

Na vstupu mějme optimalizační problém

$$\gamma^* = \max_{\boldsymbol{x} \in \mathcal{X}} H \left(\boldsymbol{x} \right) \text{ a } \boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} H \left(\boldsymbol{x} \right),$$

parametrickou rodinu hustot pravděpodobnosti $\mathbb{V} = \left\{ f \left(\cdot ; \boldsymbol{v} \right), \boldsymbol{v} \in \Theta \right\}$ definovaných na $\mathcal{X}$ a vektor počátečních pravděpodobností ASP $\boldsymbol{u}$ (například uniformní rozdělení na $\mathcal{X}$).

1. Nastavení $\boldsymbol{v}_0 = \boldsymbol{u}$, ρ a $t = 1$.
2. Spočtení γ_t:
$$\gamma_t = \max \left\{ s : \mathbb{P}_{\boldsymbol{v}_{t-1}} \left(H \left(\boldsymbol{X} \right) \geq s \right) \geq \rho \right\}.$$
3. Spočtení $\boldsymbol{v}_t$:
$$\boldsymbol{v}_t = \operatorname*{argmax}_{\boldsymbol{v} \in \Theta} \mathbb{E}_{\boldsymbol{v}_{t-1}} \left(I \left(H \left(\boldsymbol{X} \right) \geq \gamma_t \right) \cdot \ln f \left(\boldsymbol{X} ; \boldsymbol{v} \right) \right).$$
4. Pokud je splněna ukončovací podmínka (například $\gamma_t = \gamma_{t-1} = \ldots = \gamma_{t-d}$ pro pevný parametr d), ukončit algoritmus, jinak zpět k bodu 2.

Všimněme si podobnosti s algoritmem pro řešení RE úloh (algoritmus 4.2). Jedinými změnami jsou ukončovací podmínka (není zadáno pevné γ) a absence LR. CE metoda by fungovala i se zachováním LR, tedy pokud by v každém ASP byla nominální pravděpodobnost dána vektorem $\boldsymbol{u}$, ale přinášelo by to do simulace nadbytečný šum.

CE metoda pro optimalizace tedy vede k posloupnosti pravděpodobnostních rozdělení

$$f \left(\cdot ; \boldsymbol{u} \right), f \left(\cdot ; \boldsymbol{v}_1 \right), f \left(\cdot ; \boldsymbol{v}_2 \right), \ldots,$$

která bude postupně kumulovat pravděpodobnost okolo hledané hodnoty $\boldsymbol{x}^*$, bez dodatečných požadavků však není zaručena konvergence ke správné hodnotě.

Pro optimalizační úlohy, kde má přípustná množina pouze konečný počet hodnot $|\mathcal{X}| < \infty$, můžeme ukázat, že za určitých okolností CE metoda vždy dosáhne správného výsledku.

Věta 5.3 (Optimalizace pomocí CE na konečné přípustné množině)**.** *Nechť je*

- γ^* *maximum funkce* $H(\boldsymbol{x})$ *definované na konečné množině* $\mathcal{X}$*,*
- $\boldsymbol{x}^*$ *jediný parametr, pro který platí* $H(\boldsymbol{x}^*) = \gamma^*$*,*
- $\mathbb{V}$ *parametrická rodina pravděpodobnostních rozdělení* $\{f(\cdot; \boldsymbol{v}), \boldsymbol{v} \in \Theta\}$ *definovaných na* $\mathcal{X}$*, pro kterou platí, že* $\delta_{\boldsymbol{x}^*}(\boldsymbol{x}) \in \mathbb{V}$*, kde*

$$\delta_{\boldsymbol{x}^*}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} = \boldsymbol{x}^* \\ 0, & \boldsymbol{x} \neq \boldsymbol{x}^* \end{cases}$$

nazýváme atomické nebo degenerované rozdělení pravděpodobnosti v bodě $\boldsymbol{x}^*$*.*

Pak řešení CE úlohy

$$\boldsymbol{v}^* = \underset{\boldsymbol{v} \in \Theta}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{u}} \left(I\left(H(\boldsymbol{X}) \geq \gamma^*\right) \cdot \ln f(\boldsymbol{X}; \boldsymbol{v}) \right)$$

koresponduje právě s tímto $\delta_{\boldsymbol{x}^*}(\boldsymbol{x})$ *rozdělením pravděpodobnosti, čímž získáme i samotné řešení maximalizační úlohy.*

Poznámka 5.4. V případě iteračního řešení dle algoritmu 5.2 metoda konverguje k $\delta_{\boldsymbol{x}^*}(\boldsymbol{x})$ s rostoucím počtem iterací, tedy

$$\lim_{t \to \infty} f(\boldsymbol{x}, \boldsymbol{v_t}) = \delta_{\boldsymbol{x}^*}(\boldsymbol{x}).$$

Jelikož se zaměřujeme na kombinatorické optimalizační úlohy, budeme vždy množinu $\mathcal{X}$ uvažovat jako konečnou. Dále ve většině případů budeme jako parametrickou rodinu pravděpodobnosti uvažovat diskrétní pravděpodobnosti dané vektorem pravděpodobností, což jednoduše zaručí, aby $\delta_{\boldsymbol{x}^*}(\boldsymbol{x}) \in \mathbb{V}$.

Pro reálné řešení musíme jednotlivé části algoritmu 5.2 nahradit jejich SC ekvivalenty, tato implementace by však bez dodatečných úprav konvergovala ke správnému řešení jen velice obtížně, zvláště pokud bude počet pokusů N velmi malý. Toto je způsobeno možností konvergence k degenerované pravděpodobnosti v bodě jiném než $\boldsymbol{x}^*$. Například uvažujme parametrickou rodinu pravděpodobností definovanou jako

$$f(\boldsymbol{x}; \boldsymbol{u}) = \prod_{i=1}^{n} \sum_{j=1}^{m_i} I(x_i = a_{ij}) \cdot u_{ij},$$

což odpovídá diskrétní pravděpodobnosti, kde máme n nezávislých diskrétních náhodných veličin nabývajících m_i stavů. Pak při kroku metody, kdy všechny $\boldsymbol{X_1}, \boldsymbol{X}_2, ..., \boldsymbol{X}_N$ nabudou v i-té složce stejnou hodnotu, bude tato hodnota v dalších krocích metody generována s pravděpodobností jedna ($\widetilde{v}_{t,ij} = 1$). Tedy pokud $(x^*)_i \neq a_{ij}$ získali jsme špatné řešení. Poměrně jednoduchým řešením tohoto problému je využití vyhlazovacího parametru α probíraného v sekci 3.3.3.

Výsledný algoritmus je základním kamenem použití CE metody pro kombinatorické optimalizace.

Algoritmus 5.5 (Hlavní CE metoda pro optimalizace)**.**

Na vstupu mějme optimalizační problém

$$\gamma^* = \max_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}) \text{ a } \boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}),$$

parametrickou rodinu hustot pravděpodobnosti $\mathbb{V} = \{f(\cdot;\boldsymbol{v}), \boldsymbol{v} \in \Theta\}$ definovaných na $\mathcal{X}$, pro kterou $\delta_{\boldsymbol{x}^*}(\boldsymbol{x}) \in \mathbb{V}$ a vektor počátečních pravděpodobností ASP $\boldsymbol{u}$ (například uniformní rozdělení na $\mathcal{X}$).

1. Inicializace: počáteční hodnoty $\widetilde{\boldsymbol{v}}_0 = \boldsymbol{u}$, $t = 1$, ρ, N, $\alpha \in (0,1)$.
2. Vygenerování náhodných pokusů $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N \sim f(\cdot;\widetilde{\boldsymbol{v}}_{t-1})$.
3. Spočtení $H_i = H(\boldsymbol{X}_i)$, jejich seřazení a zjištění $(1-\rho)$ kvantilu pro spočtení $\gamma_t = H_{\lceil(1-\rho)\cdot N\rceil}$.
4. Spočtení $\widetilde{\boldsymbol{v}}_t$ vyřešením:

$$\widetilde{\boldsymbol{v}}_t = (1-\alpha)\cdot\widetilde{\boldsymbol{v}}_{t-1} + \alpha\cdot\operatorname*{argmax}_{\boldsymbol{v}}\left(\frac{1}{N}\sum_{i=1}^{N} I(H(\boldsymbol{X}_i)\geq\gamma_t)\cdot\ln f(\boldsymbol{X}_i;\boldsymbol{v})\right).$$

5. Pokud $\gamma_t = \gamma_{t-1} = \max H(\boldsymbol{X}_i)$ v průběhu posledních d iterací (například $d = 5$), konec algoritmu, jinak se vracíme ke kroku 2.

Implementaci algoritmu v jazyce Matlab lze nalézt v příloze, viz výpis zdrojového kódu 1.

Poznámka 5.6. Jako volby jednotlivých parametrů lze například použít $\rho = 0.01$, $\alpha = 0.7$ a $N = C\cdot R$, kde C je například 5 a R je dimenze $\boldsymbol{u}$. Velikostí hodnoty $N_b = \rho\cdot N$ budeme rozumět počet elitních vzorků, které používáme k aktualizaci vektoru parametrů $\widetilde{\boldsymbol{v}}_t$. V případě špatné konvergence lze zvyšovat hodnotu C nebo snižovat velikost vyhlazovacího parametru α, avšak obě tyto změny zvyšují výpočetní čas algoritmu.

Algoritmus 5.5 otestujeme na jednoduchém ukázkovém příkladě.

Příklad 5.7. Mějme systém o n vstupech ve tvaru binárního vektoru, např. $\boldsymbol{x} = (0,1,0,...)$, jehož optimální nastavení je dáno $\boldsymbol{y} = (0,1,0,...)$. Systémová odezva na jednotlivá nastavení (vstupní vektory) je dána funkcí:

$$H(\boldsymbol{x}) = n - \sum_{i=1}^{n} |y_i - x_i|.$$

Úkolem je nalézt neznámé optimální nastavení $\boldsymbol{y}$.

Pokud nebudeme uvažovat triviální cestu nalezení ideálního nastavení systému (postupně zkoušet vektory: $(1,0,0...)$, $(0,1,0,...)$, $(0,0,1,...)$ ...), roste složitost řešení takovéhoto problému exponenciálně. Tedy pro n vstupů je třeba v nejhorším případě vyzkoušet 2^n vstupních vektorů.

K řešení této úlohy použijeme algoritmus 5.5.

Jako testovací úlohu volíme skrytý vektor délky 100. Počáteční rozdělení pravděpodobnosti (v případě předchozích úloh nominální rozdělení) je obecně volitelné. V případě, že o parametrech nemáme žádné bližší informace, je vhodné volit počáteční rozdělení tak, aby nepreferovalo žádný stavový vektor. Jako vstupní parametry CE metody tedy použijeme:

$$\alpha = 0.9, N = 500, \rho = 0.1, \boldsymbol{p} = (0.5, 0.5, ..., 0.5)\,.$$

Řešení úlohy provedeme vícekrát, výstupní hodnoty lze nalézt v následující tabulce.

	1. běh	2. běh	3. běh	4. běh	5.běh
Počet iterací	8	7	8	8	8
Výkon systému na konci v procentech	100%	100%	100%	100%	100%

Tabulka 5.1: Výsledky ukázkové úlohy $n = 100$

Pro nalezení optimálního nastavení jsme potřebovali v průměru přibližně 4000 pokusů, při analytickém řešení bychom museli projít všech $1.2677 \cdot 10^{30}$ možností.

Dále uvedeme výsledky pro rozsáhlejší úlohu: $n = 1000, N = 5000$.

	1. běh	2. běh	3. běh	4. běh	5.běh
Počet iterací	26	27	26	26	26
Výkon systému na konci v procentech	100%	100%	100%	100%	100%

Tabulka 5.2: Výsledky ukázkové úlohy $n = 1000$

△

V této úloze jsme vždy nalezli optimální řešení, což obecně nemusí být pravda, složitější úlohou se budeme zabývat v příkladě 5.11.

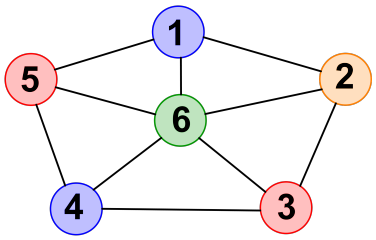
Většina kombinatorických úloh lze vyjádřit pomocí stochastické sítě řízené vrcholy (max-cut problém) nebo hranami (TSP). V dalších částech ukážeme obecné formulace těchto úloh s postupem aktualizace parametrů.

5.2 Aktualizace parametrů ve stochastické síti dané vrcholy

Stochastická síť daná vrcholy (*„stochastic node network"* dále jen SNN) může být definována následovně.

Definice 5.8 (SNN). Mějme graf $G = (V, E)$ daný množinou vrcholů $V = \{v_i\}_{i=1}^n$ o velikosti n a množinou hran $E = \{e_i\}_{i=1}^r$ o velikosti r. Dále mějme vektor $\boldsymbol{x} = (x_1, \ldots, x_n)$ přiřazující jednotlivým vrcholům i stavy x_i z množiny stavů Ξ (například barvení grafu, kde Ξ je množina možných barev). Pokud jsou jednotlivé stavy vrcholů dány nějakým rozdělením pravděpodobnosti definovaném na Ξ, jedná se o SNN.

Příkladem SNN je barvení vrcholů grafu viz obrázek 5.1.



Obrázek 5.1: Příklad SNN (barvení vrcholů grafu)

Pro optimalizační úlohy budeme dále předpokládat, že každý vektor stavů vrcholů $\boldsymbol{x} = (x_1, \ldots, x_n)$ má ohodnocení $H(\boldsymbol{x})$, které chceme maximalizovat (případně minimalizovat). Množinou stavů $\mathcal{X}$ budeme rozumět všechny kombinace stavů vrcholů, tedy máme-li m stavů vrcholů, je množina $\mathcal{X} = \{a_1, \ldots, a_m\}^n$.

Nejednodušším způsobem, jak převést optimalizační úlohu

$$\max_{\boldsymbol{x} \in \mathcal{X}} H(\boldsymbol{x})$$

na ASP, je použít sdružené diskrétní rozdělení pravděpodobnosti

$$f(\boldsymbol{x}, \boldsymbol{p}) = \prod_{i=1}^{n} \sum_{j=1}^{m} I(x_i = a_j) \cdot p_{ij}.$$

Tedy jednotlivá rozdělení stavů vrcholů jsou nezávislá. Jako parametrickou rodinu pravděpodobnosti budeme v těchto úlohách považovat $\mathbb{V} = \{f(\cdot; \boldsymbol{p}), \boldsymbol{p} \in \Theta\}$, kde jednotlivé řádky matice $\boldsymbol{p}$ splňují podmínky diskrétních pravděpodobností (nezápornost prvků a jednotkový součet). Použitím znalostí z věty 3.26 můžeme jednoduše vyjádřit formule pro aktualizaci parametrů jako

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} \left(I(H(\boldsymbol{X}) \geq \gamma_t) \cdot I(X_i = a_j) \right)}{\mathbb{E}_{\boldsymbol{p}_{t-1}} \left(I(H(\boldsymbol{X}) \geq \gamma_t) \right)}$$

a v případě řešení pomocí SC jako

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I(H(\boldsymbol{X}_k) \geq \gamma_t) \cdot I(X_{ki} = a_j)}{\sum_{k=1}^{N} I(H(\boldsymbol{X}_k) \geq \gamma_t)}. \tag{5.1}$$

5.2.1 Podmíněné generovaní vzorků

V mnoha případech SNN úloh nebudeme chtít optimalizovat přes celou množinu přípustných stavů $\mathcal{X}$, ale pouze přes nějakou podmnožinu $\mathcal{Y} \subset \mathcal{X}$. Například v případě max-cut problému můžeme chtít pouze řezy, které dělí množinu vrcholů na stejně velké části. Tento požadavek vyžaduje změnu generování vzorků společně se změnou aktualizační formule parametrů rozdělení.

Je zřejmé, že problém

$$\max_{\boldsymbol{x} \in \mathcal{Y}} H(\boldsymbol{x})$$

lze zapsat také jako

$$\max_{\boldsymbol{x} \in \mathcal{X}} H^*(\boldsymbol{x}),$$

kde $H^*(\boldsymbol{x}) = H(\boldsymbol{x})$ pro $\boldsymbol{x} \in \mathcal{Y}$ a $H^*(\boldsymbol{x}) = -\infty$ pro $\boldsymbol{x} \notin \mathcal{Y}$. Pro generování snímku pak můžeme použít *„acceptance-rejection"*, kde budeme zahazovat snímky nepatřící do $\mathcal{Y}$. V tomto případě zůstane aktualizační formule shodná s 5.1. V případě analytického řešení získáme tvar aktualizační formule jako

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}}\left(I\left(H\left(\boldsymbol{X}\right) \geq \gamma_t\right) \cdot I\left(X_i = a_j\right) | \boldsymbol{X} \in \mathcal{Y}\right)}{\mathbb{E}_{\boldsymbol{p}_{t-1}}\left(I\left(H\left(\boldsymbol{X}\right) \geq \gamma_t\right) | \boldsymbol{X} \in \mathcal{Y}\right)}.$$

5.2.2 Degenerované rozdělení pravděpodobnosti

Předpokládejme, že řešení SNN maximalizačního problému

$$H(\boldsymbol{x}^*) = \gamma^* = \max_{\boldsymbol{x} \in \mathcal{X}} H(\boldsymbol{x})$$

je jediné. Potom parametr $\boldsymbol{p}^*$ optimálního/degenerovaného rozdělení pravděpodobnosti má tvar

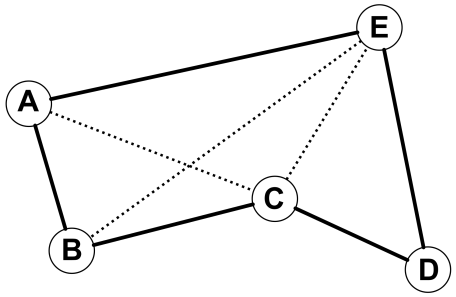
$$p_{ij}^* = \begin{cases} 1, & x_i^* = a_j \\ 0, & x_i^* \neq a_j \end{cases}.$$

5.3 Aktualizace parametrů ve stochastické síti dané hranami

Stochastická síť daná hranami (*„stochastic edge network"* dále jen SEN) může být definována následovně.

Definice 5.9 (SEN)**.** Mějme graf $G = (V, E)$ daný množinou vrcholů $V = \{v_i\}_{i=1}^n$ o velikosti n a množinou hran $E = \{e_i\}_{i=1}^r$ o velikosti r. Dále mějme vektor z vícerozměrného Bernoulliho rozdělení $\boldsymbol{x} = (x_1, \ldots, x_n), x_i \in \{0, 1\}$ o velikosti r, který udává výběr z množiny hran. Těmto úlohám, kde výskyt jednotlivých hran podléhá Bernoulliho rozdělení, budeme říkat SEN.

Příkladem SEN je TSP viz obrázek 5.2.



Obrázek 5.2: Příklad SEN (TSP)

Jednoduchým přístupem ke generování pokusů z $\mathcal{X} = \{0,1\}^n$ je generování jednotlivých x_i z Bernoulliho rozdělení. V reálných úlohách však často potřebujeme hledat cesty grafu G (cestou grafu budeme dále rozumět vektor posloupnosti navštívených vrcholů), které splňují nějaké vlastnosti (například TSP). Pro tento typ úloh se jako vhodný přístup jeví využití Markovových řetězců (MŘ).

Dále odvoďme tvar aktualizačních formulí při generování posloupnosti $X_1, \ldots, X_m$ vrcholů. Pro jednoduchost budeme předpokládat, že začátek MŘ je fixní, například 1. vrchol a pravděpodobnost přechodu je dána přechodovou maticí $\boldsymbol{P} = (p_{ij})$, kde

$$p_{ij} = \mathbb{P}\left(X_{k+1} = j | X_k = i\right), \; k = 1, \ldots, m-1.$$

Logaritmus hustoty pravděpodobnosti $f\left(\boldsymbol{x}, \boldsymbol{P}\right)$ se dá vyjádřit jako

$$\ln f\left(\boldsymbol{x}, \boldsymbol{P}\right) = \sum_{r=1}^{m} \sum_{i,j} I\left(\boldsymbol{x} \in \mathcal{X}_{ij}\left(r\right)\right) \cdot \ln p_{ij},$$

kde $\mathcal{X}_{ij}\left(r\right)$ je množina všech cest v $\mathcal{X}$, které v r-tém přechodu přecházejí z vrcholu i do vrcholu j, viz [1, str. 139]. Pro funkci $S\left(\boldsymbol{x}\right) = I\left(H\left(\boldsymbol{x}\right) \geq \gamma\right)$ lze jednoduše odvodit formuli pro analytickou aktualizaci parametru

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{P}_{t-1}} I\left(H\left(\boldsymbol{X}\right) \geq \gamma_t\right) \cdot \sum_{r=1}^{m} I\left(\boldsymbol{X} \in \mathcal{X}_{ij}\left(r\right)\right)}{\mathbb{E}_{\boldsymbol{P}_{t-1}} I\left(H\left(\boldsymbol{X}\right) \geq \gamma_t\right) \cdot \sum_{r=1}^{m} I\left(\boldsymbol{X} \in \mathcal{X}_{i}\left(r\right)\right)},$$

kde $\mathcal{X}_{i}\left(r\right)$ je množina všech cest v $\mathcal{X}$, které v r-tém přechodu přecházejí z vrcholu i. Příslušná aktualizační formule pro SC je pak dána

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \geq \gamma_t\right) \cdot \sum_{r=1}^{m} I\left(\boldsymbol{X}_k \in \mathcal{X}_{ij}\left(r\right)\right)}{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \geq \gamma_t\right) \cdot \sum_{r=1}^{m} I\left(\boldsymbol{X}_k \in \mathcal{X}_{i}\left(r\right)\right)}.$$

5.3.1 Podmíněné generovaní vzorků

Stejně jako u SNN úloh můžeme v SEN úlohách chtít optimalizovat pouze přes nějakou podmnožinu všech přípustných stavů $\mathcal{Y} \subset \mathcal{X}$. Příkladem takovéhoto omezení je požadavek na navštívení každého vrcholu právě jednou (TSP).

Toto lze vyřešit stejně jako u SNN úloh pomocí zahazování nevhodných snímků a úpravou výkonostní funkce. Opět nedojde ke změně aktualizační formule pro SC úlohy.

Jedním z častých omezení je simulace MŘ s nahrazováním (*„Markov chain with replacement"* dále jen MCWR), kde vylučujeme navštívení jednoho vrcholu vícekrát. Toto provedeme tak, že v každém kroku vynulujeme sloupec matice přechodu patřící navštívenému stavu a matici přechodu znormalizujeme. Je zřejmé, že se již formálně nejedná o MŘ, toto však na simulaci nemá vliv.

5.3.2 Degenerované rozdělení pravděpodobnosti

Předpokládejme jediné optimální řešení γ^* s argumentem $\boldsymbol{x}^*$ a použití každého vrcholu maximálně jednou. Pak pro matici přechodu degenerovaného rozdělení platí: $p_{ij}^* = 1$, pokud se ve vektoru $\boldsymbol{x}^*$ nachází přechod z i do j, ostatní prvky v daném řádkou jsou nulové. Ostatní řádky jsou libovolné, neboť se do stavu, který zastupují, nelze dostat.

5.4 Plně adaptivní CE algoritmus pro optimalizace

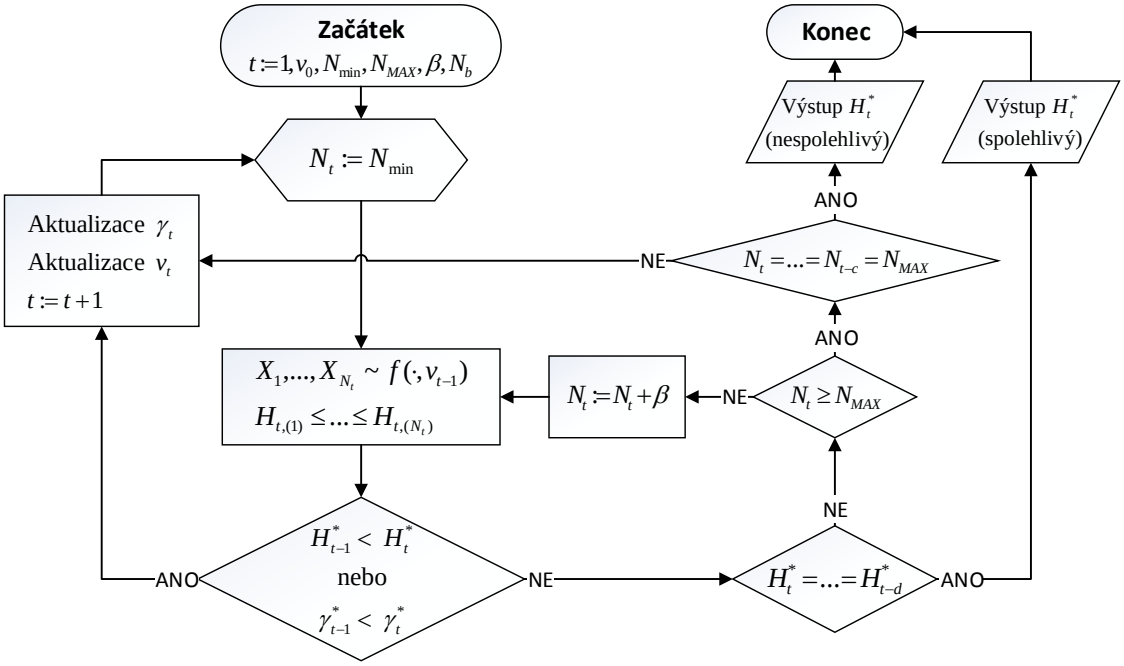
Jako alternativu k CE metodě uveďme modifikaci algoritmu 5.5, plně adaptivní CE algoritmus (*„fully adaptive CE"*, dále jen FACE). Tato modifikace se týká adaptivního zvyšování počtu pokusů v případě stagnace metody a odhalení špatně řešitelných problémů (nestabilní řešení). [1, strana 194]

Algoritmus 5.10 (FACE algoritmus)**.**

---

1. Inicializace: $\widetilde{\boldsymbol{v}}_0, N_b, N_{min}, N_{MAX}, d, c, \beta, \alpha$ a $t = 1$.
2. Nastavení $N_t = N_{min}$.
3. Vygenerování vzorků $X_1, \ldots, X_{N_t} \sim f(\cdot; \widetilde{\boldsymbol{v}}_{t-1})$, spočítání a seřazení jejich výkonnostních funkcí $H_{t,(1)}, \ldots, H_{t,(N_t)}$ a nastavení $\gamma_t = H_{t,(N_t - N_b)}, H_t^* = H_{t,(N_t)}$.
4. Pokud platí $\gamma_{t-1} < \gamma_t$ nebo $H_{t-1}^* < H_t^*$, pak aktualizovat parametry γ_t a $\widetilde{\boldsymbol{v}}_t$, inkrementovat $t = t + 1$ a pokračovat krokem 2, pokud neplatí, pokračovat krokem 5.
5. Pokud platí $H_t^* = \ldots = H_{t-d}^*$, ukončit algoritmus se spolehlivým výsledkem H_t^*, pokud neplatí, pokračovat krokem 6.
6. Pokud platí $N_t < N_{MAX}$, zvýšit počet pokusů $N_t = \min\{N_{max}, N_t + \beta\}$ a pokračovat krokem 3. (generují se však pouze snímky navíc, tedy maximálně β), pokud neplatí, pokračovat krokem 7.
7. Pokud platí $N_t = \ldots N_{t-c} = N_{MAX}$, ukončit algoritmus s nespolehlivým výsledkem H_t^*, pokud neplatí, pokračovat krokem 4.

Pro přehlednost uveďme ještě diagram průběhu algoritmu.



Obrázek 5.3: Diagram průběhu FACE algoritmu

Implementaci FACE algoritmu v jazyce Matlab lze nalézt v příloze, viz výpis kódu 2. Tento algoritmus vyzkoušíme na následujícím příkladě.

Příklad 5.11 (Problém umístění n královen)**.** Mějme dánu šachovnici velikosti $n \times n$ a n šachových figurek královny. Problémem je nalézt umístění těchto královen tak, aby žádná figura neohrožovala jinou figuru (jedna figurka královny ohrožuje řádek, sloupec a obě diagonály, ve kterých je umístěna).

Pro tuto úlohu budeme uvažovat dvě různé ASP formulace:

1. Jednotlivé královny jsou umístěny na různých řádcích, my tedy generujeme sloupcové pozice jednotlivých královen. Jedná se tedy o SNN s n vrcholy o n stavech.

2. Jednotlivé královny musí mít řádkové i sloupcové pozice unikátní, proto lze jednotlivé kandidáty vybírat pouze z permutací n čísel. Toto budeme simulovat pomocí SEN simulované jako MŘ, kde počet kroků MŘ bude n a budeme vyžadovat MCWR (MŘ bez opakování vrcholů). Počáteční matici přechodu pro tuto úlohu sestavíme následovně:

   (a) Vytvoříme matici samých jedniček o velikosti $n + 1$

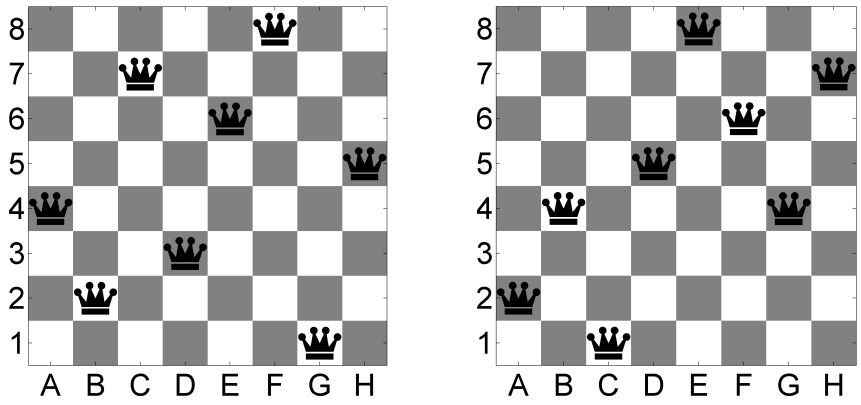
   (b) Vynulujeme první sloupec a snížíme hodnoty v třech hlavních diagonálách na 0.01 (mimo hodnoty v prvním řádku), tato druhá úprava sníží počáteční šanci, že královna umístěná v k-tém kroce ohrozí královnu v $k - 1$ kroce.

(c) Normalizujeme vůči řádkovému součtu.

Jako fixní začátek volíme stav 1, přičemž tento stav je uměle přidán a reprezentuje pravděpodobnosti pro pozici první královny.

Jako výkonnostní funkci budeme uvažovat n mínus počet špatně umístěných královen, tedy počet vzájemných ohrožení (například jsou-li tři královny v jednom sloupci bude výkonostní funkce o 2 menší než maximum). Maximum výkonnostní funkce je tedy $\gamma^* = n$. Je zřejmé, že druhý přístup bude nejspíše efektivnější, neboť $|\mathcal{X}| = n^n$ v případě první formulace a $|\mathcal{X}| = n!$ v případě druhé formulace, což je mnohem menší počet přípustných stavů.

Ukázka optimálního a neoptimálního řešení pro $n = 8$ lze vidět na obrázku níže.



(a) $H(x) = 8$ (b) $H(x) = 6$

Obrázek 5.4: Problém umístění $n = 8$ královen

Nejprve budeme řešit úlohu pro $n = 8$, tedy standardní šachovnici. Pro porovnání použijeme CE a FACE aplikované na obě ASP formulace úlohy, parametry algoritmu FACE volíme

$$\alpha = 0.5, d = 5, c = 3, \rho = 0.05$$

přičemž parametry $N_{min}, N_{MAX}, \beta, N_b$ záleží na typu ASP. Obecně, má-li matice aktualizovaných parametrů počet prvků k, volíme $N_{min} = k, N_{MAX} = 20 \cdot N_{min}, \beta = N_{min}, N_b = \lceil \rho \cdot k \rceil$. V případě CE algortimu volíme parametry

$$\alpha = 0.5, d = 5, \rho = 0.05, N = k \cdot 5.$$

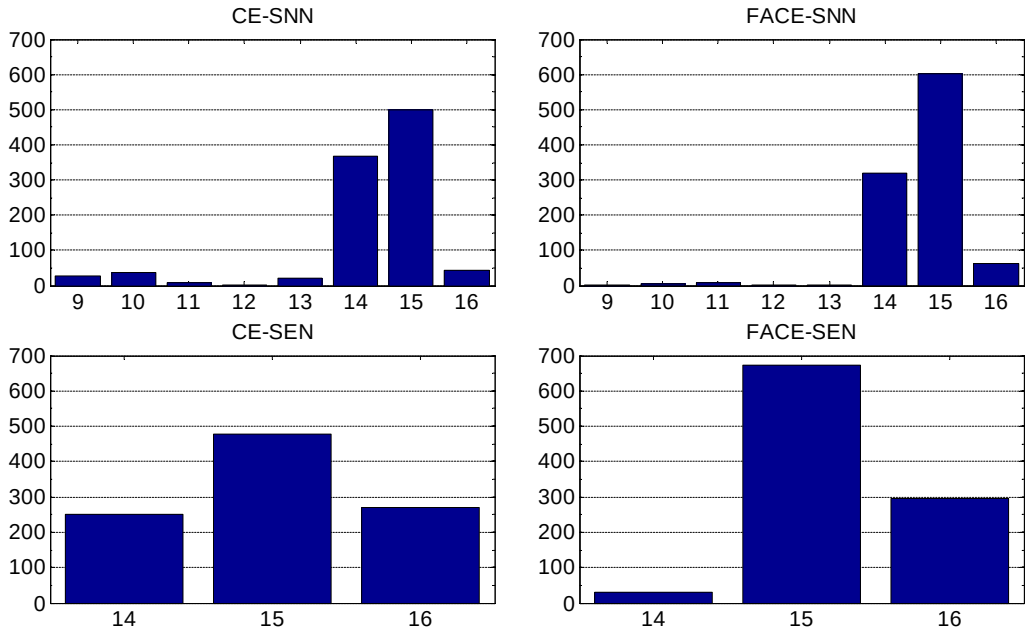
Pro věrohodné porovnání obou algoritmů a přístupů budeme úlohu řešit 1000 krát. Výsledky testu lze vidět v tabulce 5.3, sledovanými hodnotami je průměr času výpočtu $\overline{čas}$, maximální čas výpočtu $\max čas$, průměrný celkový počet použitých pokusů $\overline{N_{sum}}$, maximální celkový počet použitých pokusů $\max N_{sum}$, průměrný počet iterací $\overline{Iter}$, maximální počet iterací $\max Iter$, průměrný nalezený maximální výkon $\overline{H^*}$ a minimum z nalezených maximálních výkonů $\min H^*$.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	min H^*
CE-SNN	0.008	0.03	2834	5120	8.9	16	7.26	6
CE-SEN	0.008	0.072	2026	2430	5	8	8	8
FACE-SNN	0.031	0.067	5543	12544	10.3	17	7.37	6
FACE-SEN	0.037	0.048	5365	7128	6.3	8	8	8

Tabulka 5.3: Výsledek testování problému umístění královen $(n = 8)$

Jak se dalo očekávat, ASP formulace pomocí SEN vykazuje mnohem lepší výsledky. V případě srovnání CE a FACE algoritmu lze sledovat u FACE algoritmu mnohem vyšší celkový počet pokusů. Pro lepší porovnání obou algoritmů zkusíme ještě otestovat úlohu s $n = 16$ (jedno z řešení lze nalézt v příloze, obrázek D.1).

Výsledky testu pro 1000 simulací lze nalézt v tabulce 5.4 a rozdělení nalezených hodnot v obrázku 5.5.



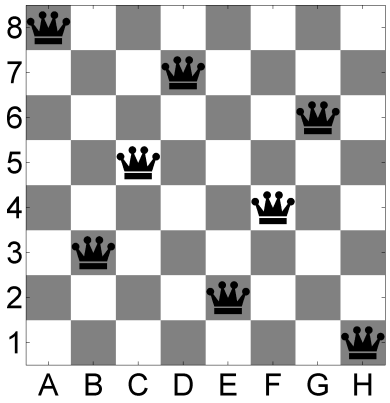
Obrázek 5.5: Histogramy jednotlivých řešení problému umístění $n = 16$ královen

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	min H^*
CE-SNN	0.065	0.097	21363	32000	16.7	25	14.3	9
CE-SEN	0.114	0.323	21450	60690	14.8	42	15.0	14
FACE-SNN	0.167	0.437	36098	66560	18.9	26	14.7	9
FACE-SEN	0.338	0.766	38348	85255	10.1	20	15.3	14

Tabulka 5.4: Výsledek testování problému umístění královen $(n = 16)$

Z porovnání na větší úloze vidíme, že FACE algoritmus sice využívá více pokusů, avšak častěji konverguje ke skutečnému řešení úlohy.

Jak lze vidět, úloha umístnění n královen je pro velká n jen obtížně řešitelná. Toto je dáno složitou výkonnostní funkcí, která má mnoho lokálních maxim. Příkladem je umístnění královen, kdy pouze jedna královna ohrožuje jinou avšak pro optimální umístnění je třeba přemístit více královen (ne pouze jednu) viz obrázek 5.6.



Obrázek 5.6: Lokální maximum v problému umístění $n = 8$ královen

△

5.5 Konvergence CE metody pro optimalizace

Na závěr této kapitoly uveďme několik vět zabývajících se konvergencí CE metody pro optimalizace. Budeme se zabývat pouze kombinatorickou optimalizací na prostoru přípustných stavů $\mathcal{X} = \{0,1\}^n$, toto však není přílišné omezení neboť většina kombinatorických problémů lze na tuto jednoduchou variantu převést.

Jako algoritmus, jehož konvergenci zkoumáme volíme algoritmus 5.5 s proměnným parametrem $\{\alpha_t\}_{t=1}^{\infty}$. Vektorem počátečních parametrů budeme rozumět pravděpodobnosti $p_i = \mathbb{P}(X_i = 1) = 0.5$.

Nalezením optimálního řešení v následujících větách budeme rozumět vygenerování vzorku $X^* : H(X^*) = \gamma^*$, přičemž řešení nemusí být právě jedno. Ve všech případech se bude jednat o konvergenci s $t \to \infty$.

Veškeré věty a poznámky, které zmíníme, lze najít včetně důkazů v [6].

Věta 5.12 (Nutná podmínka konvergence algoritmu 5.5. [6])**.** *Algoritmus 5.5 nalezne optimální řešení s pravděpodobností 1 jen tehdy, když posloupnost $\{a_t\}_{t=1}^{\infty}$ splňuje podmínku*

$$\sum_{t=1}^{\infty} \prod_{m=1}^{t} (1 - \alpha_m) = \infty.$$

Věta 5.13 (Postačující podmínka konvergence algoritmu 5.5. [6]). *Jestliže posloupnost* $\{a_t\}_{t=1}^{\infty}$ *splňuje podmínku*

$$\sum_{t=1}^{\infty} \prod_{m=1}^{t} (1-\alpha_m)^n = \infty,$$

pak algoritmus 5.5 nalezne optimální řešení s pravděpodobností 1.

Poznámka 5.14. Postačující podmínka platí také při posloupnostech vyhlazovacích parametrů $\{a_t\}_{t=1}^{\infty}$ splňujících podmínku $\sum_{t=1}^{\infty} \alpha_t < \infty$. [6]

Poznámka 5.15. Pro dané parametry $N, \{a_t\}_{t=1}^{\infty}, \boldsymbol{p}_0$ můžeme spočítat dolní mez pravděpodobnosti, že v T iteracích nalezneme optimální řešení. [6]

Z těchto vět je například patrné, že při použití konstantního vyhlazovacího parametru α nemáme nikdy zajištěnou konvergenci k optimálnímu řešení. Při použití konstantního vyhlazovacího parametru je velice zajímavá následující věta.

Věta 5.16 (Konvergence algoritmu 5.5 s konstantním vyhlazovacím parametrem. [6]). *Algoritmus 5.5 s konstantním vyhlazovacím parametrem* α *konverguje k degenerované pravděpodobnosti* δ_y *v nějakém bodě* $y \in \mathcal{X}$. *Navíc platí, že vhodnou volbou parametru* α *lze zařídit, aby pravděpodobnost nalezení optimálního řešení byla libovolně blízko 1.*

Věta 5.16 tedy říká, že při použití konstantního vyhlazovacího parametru lze vylepšit špatnou konvergenci metody pomocí jeho snížením, což v v případě některých optimalizačních problémů použijeme.

6 Aplikace CE metody na vybrané úlohy kombinatorických optimalizací

V této části ukážeme řešení několika základních problémů kombinatorické optimalizace, konkrétně max-cut problému, problému obchodního cestujícího a problému zarovnání sekvencí. Na závěr pak uvedeme srovnání CE metody s jinými dostupnými simulačními metodami pro řešení kombinatorických optimalizací na problému obchodního cestujícího.

6.1 Max-cut problém

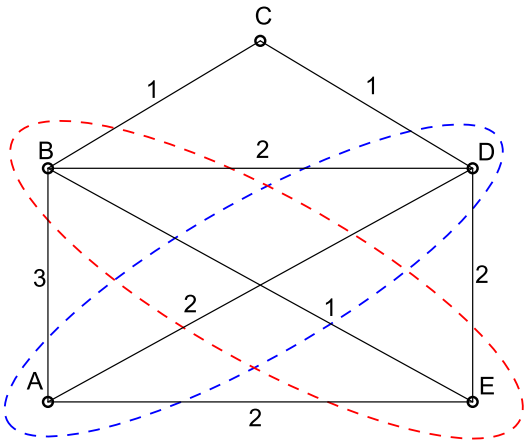
Max-cut problémem označujeme úlohu na hledání maximálního řezu grafu, čerpáme především z [1]. Řezem grafu rozumíme rozdělení vrcholů grafu do dvou neprázdných množin. Maximálním řezem pak nazveme řez takový, pro který je součet všech vah hran mezi těmito množinami největší.

Pro snadnější porozumění úloze uveďme krátký ukázkový příklad.

Příklad 6.1 (Ukázková úloha)**.** Mějme neorientovaný graf s kladně ohodnocenými hranami daný následující maticí sousednosti.

	A	**B**	**C**	**D**	**E**
A	0	3	0	2	2
B	3	0	1	2	1
C	0	1	0	1	0
D	2	2	1	0	2
E	2	1	0	2	0

Tabulka 6.1: Max-cut ukázkový příklad: Matice sousednosti



Obrázek 6.1: Max-cut ukázkový příklad: Graf s kladně ohodnocenými cestami

Řešení zřejmě nalezneme, pokud projdeme všechny kombinace rozdělení vrcholů grafu do dvou množin V_1 a V_2. Těchto dělení je však 2^n, kde n je počet vrcholů. V praxi však stačí ověřit 2^{n-1} dělení neboť dělení $V_1 = \{A, B, C\}, V_2 = \{D, E\}$ a $V_1 = \{D, E\}, V_2 = \{A, B, C\}$ má stejnou hodnotu řezu.

Řešení s grafickou reprezentací grafu můžeme na obrázku 6.1. △

6.1.1 Řešení Max-cut problému CE metodou

V této části ukážeme použití CE metody pro nalezení maximálního řezu grafu. Uvažujme pouze neorientované kladně ohodnocené grafy. Jelikož přiřazujeme charakteristiky jednotlivým vrcholům, jedná se o SNN.

Bez újmy na obecnosti uvažujme úplný graf, máme tedy množinu vrcholů $V = \{v_i\}$ a množinu kladných ohodnocení hran (cest) $C = \{c_i\}$ (v úplném grafu každou dvojici vrcholů spojuje cesta). Množinu cest $C = \{c_i\}$ budeme dále ztotožňovat s maticí vah jednotlivých cest $C = (c_{ij})$, kde c_{ij} je váha cesty mezi vrcholy v_i a v_j.

Rozdělení vrcholů do jednotlivých množin V_1 a V_2 lze jednoduše realizovat pomocí vícerozměrného Bernoulliho rozdělení, přičemž 1 znamená příslušnost k množině V_1 a 0 příslušnost k množině V_2. Vektorem parametrů jsou tedy pravděpodobnosti příslušnosti jednotlivých vrcholů k množinám V_1 a V_2.

Jelikož máme vícerozměrné diskrétní rozdělení pravděpodobnosti s konečným počtem stavů, můžeme dle předchozích poznatků zapsat aktualizační formule parametrů jako:

$$\widetilde{p}_{t,i} = \frac{\sum_{j=1}^{N} I\left(H\left(\boldsymbol{X}_j\right) \geq \gamma_t\right) \cdot I\left(X_{j,i} = 1\right)}{\sum_{j=1}^{N} I\left(H\left(\boldsymbol{X}_j\right) \geq \gamma_t\right)}.$$

Jako výkonostní funkci zřejmě použijeme váhu vygenerovaného řezu

$$H\left(\boldsymbol{x}\right) = \sum_{v_i \in V_1, v_j \in V_2} c_{i,j}, \;\; \boldsymbol{x} = (x_1, \ldots, x_n) : \begin{cases} x_i = 1 & \Rightarrow v_i \in V_1 \\ x_i = 0 & \Rightarrow v_i \in V_2 \end{cases}.$$

Poznámka 6.2. Je zřejmé, že vždy existují alespoň dvě optimální řešení (záměna V_1 a V_2). Toto lze řešit zafixováním příslušnosti jednoho vrcholu k dané množině, například

$$\widetilde{p}_{0,1} = 1 \rightarrow v_1 \in V_1.$$

Toto však vytvoří z předchozí úlohy úlohu novou, která obsahuje významné lokální maximum. Numerické testy ukázaly, že tato úprava vede častěji k neoptimálnímu řešení.

Syntetická úloha Jelikož se jedná o problém, který je pro větší rozměry úlohy analytickou cestou neřešitelný, musíme pro možnost ověření správného výsledku použít uměle vytvořenou úlohu, kde budeme znát optimální řešení. Předem rozdělíme vrcholy do dvou stejně velikých množin V_1 a V_2 (potřebujeme tedy $n = 2m$ vrcholů). Úloha bude mít následující vlastnosti:

- váha každé cesty mezi vrcholy v rámci stejné množiny bude < 1,
- váha cesty mezi vrcholy z různých množin bude 1.

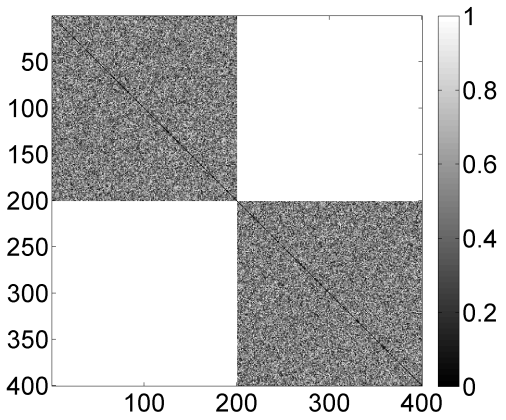
Uspořádáme-li váhy cest do matice, tak že váha na pozici i, j bude spojovat v_i s v_j a $V_1 = \{v_1, ..., v_m\}, V_2 = \{v_{m+1}, ..., v_{2m}\}$, získáme matici vah následujících vlastností:

- čtvercová,
- symetrická,
- na diagonále jsou 0,
- je tvořena bloky:

$$C = \begin{pmatrix} Z_{11} & B_{12} \\ B_{21} & Z_{22} \end{pmatrix}, B_{12} = B_{21} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix},$$

$$(Z_{11})_{ij} = \begin{cases} x \sim \mathcal{U}(0,1), & i < j \\ (Z_{11})_{ji}, & j > i \\ 0, & i = j \end{cases}, (Z_{22})_{ij} = \begin{cases} x \sim \mathcal{U}(0,1), & i < j \\ (Z_{22})_{ji}, & j > i \\ 0, & i = j \end{cases}.$$

Ukázku takovéto vygenerované matice lze vidět na následujícím obrázku.



Obrázek 6.2: Ukázka použité matice vah $2 \cdot m = 400$

Jako testovací úlohu zvolíme počet vrcholů $n = 2m = 400$. Použijeme CE a FACE algoritmus s nastavením parametrů

- FACE: $\alpha = 0.5, d = 5, c = 3, \rho = 0.05, N_{min} = n, N_{MAX} = 20 \cdot N_{min}, \beta = N_{min}, N_b = \lceil \rho \cdot n \rceil$,

- CE: $\alpha = 0.5, d = 5, \rho = 0.05, N = n \cdot 5, N_b = \lceil \rho \cdot n \rceil$.

Maximální řez této syntetické úlohy má váhu $200^2 = 40000$. Výsledky 100 běhů programu lze vidět v následující tabulce, sledovanými hodnotami je průměr času výpočtu $\overline{čas}$, maximální čas výpočtu $\max čas$, průměrný celkový počet použitých pokusů $\overline{N_{sum}}$, maximální celkový počet použitých pokusů $\max N_{sum}$, průměrný počet iterací $\overline{Iter}$, maximální počet iterací $\max Iter$, průměrný nalezený maximální výkon $\overline{H^*}$ a minimum z nalezených maximálních výkonů $\min H^*$.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\min H^*$
CE	4.64	5.28	67700	76000	33.9	38	40000	40000
FACE	2.7	7.82	40132	106400	32.7	50	38252	30331

Tabulka 6.2: Výsledky řešení syntetické max-cut úlohy pro $n = 400$ (100 běhů)

Jak lze vidět z výsledků řešení, CE algoritmus využívá více více pokusů, avšak v každém ze 100 běhů programu nalezl optimální řešení.

Dále otestujeme CE a FACE algoritmus pro rozsáhlejší úlohu $2 \cdot m = n = 1000$, maximální řez má v tomto případě váhu $500^2 = 250000$. Výsledky jsou uvedeny v tabulce níže, jedná se o údaje získané 10 běhy algoritmů.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\min H^*$
CE	220	240	263000	285000	52.6	57	250000	250000
FACE	102	339	123300	373000	50.5	54	243920	189250

Tabulka 6.3: Výsledky řešení syntetické max-cut úlohy pro $n = 1000$ (10 běhů)

Jak lze vidět z výsledků CE algoritmus i pro takto rozsáhlou úlohy vždy nalezl optimální řešení. Déle lze pozorovat rostoucí časy požadované pro vyřešení úlohy, což je motivací pro paralelní implementaci viz sekce 8.3.1.

6.1.2 Max-cut s více podmnožinami

Přirozeným rozšířením max-cut problému je dělení množiny vrcholů na více než dvě množiny. V tomto případě získáme téměř shodný problém jako v případě SNN ASP pro problém umístění n královen (viz příklad 5.11), jediným rozdílem bude výkonnostní funkce.

Máme tedy k množin $V_1, \ldots, V_k$, do kterých chceme rozdělit n vrcholů tak, aby součet cest mezi jednotlivými množinami byl největší. Jednotlivé příslušnosti vrcholů k množinám dělení budeme generovat jako vícerozměrné diskrétní rozdělení pravděpodobnosti s konečným počtem stavů. Vektorem počátečních parametrů bude tedy matice s řádkovými součty rovnými jedné a o velikosti $n \times k$. Aktualizační formule pro parametry rozdělení bude

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \geq \gamma_t\right) \cdot I\left(X_{k,i} = j\right)}{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \geq \gamma_t\right)}.$$

A výkonnostní funkce bude definována jako

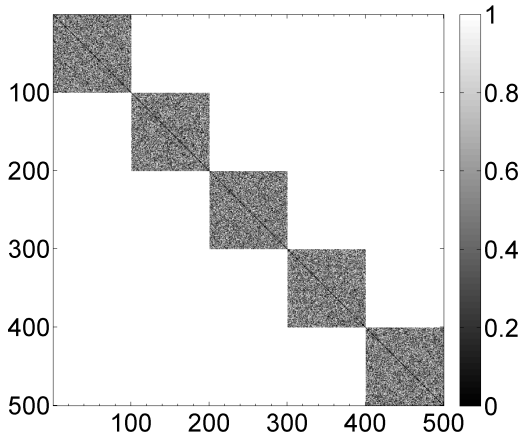
$$H(\boldsymbol{x}) = \sum_{a=1}^{k} \sum_{b=a+1}^{k} \sum_{v_i \in V_a, v_j \in V_b} c_{i,j},\ \boldsymbol{x} = (x_1, \ldots, x_n) : x_i = a \Rightarrow v_i \in V_a.$$

Pro ověření správného řešení budeme opět uvažovat syntetický problém, kde předem vytvoříme k stejně velkých množin a zajistíme, aby cesty mezi vrcholy jedné množiny byly $c_{ij} < 1$ a mezi vrcholy různých množin $c_{ij} = 1$. Konstrukci matice vah provedeme stejným způsobem jako v případě syntetické úlohy dělení na dvě množiny.

Jako testovací úlohu volíme počet množin $k = 5$ a počet vrcholů $n = 500$. Maximum výkonnostní funkce je tedy

$$\gamma^* = (n/k)^2 \cdot \sum_{i=1}^{k-1} i = 10000 \cdot 10 = 10^5.$$

Ukázku matice vah pro tuto syntetickou úlohu lze vidět na následujícím obrázku.



Obrázek 6.3: Ukázka použité matice vah $n = 500, k = 5$

Výsledky řešení pomocí CE a FACE algoritmu lze vidět v následující tabulce, z důvodu velké časové náročnosti je provedeno pouze 10 běhů algoritmů.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	min H^*
CE	369	413	935000	1050000	74.8	84	100000	100000
FACE	177	419	415750	970000	21.4	71	91211	90212

Tabulka 6.4: Výsledky řešení syntetické max-cut úlohy pro 500 vrcholů a 5 množin

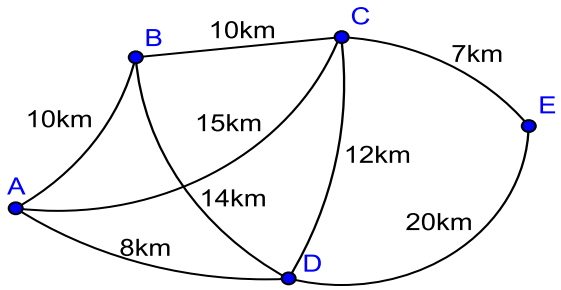
CE algoritmus vyřešil úlohu ve všech 10 bězích, FACE algoritmus pouze v jednom případě.

6.2 Problém obchodního cestujícího

Problémem obchodního cestujícího (*„traveling salesman problem"* dále jen TSP) budeme rozumět úlohu na hledání nejkratšího hamiltonovského cyklu grafu. Jako vstupní data úlohy budeme uvažovat pouze úplné neorientované kladně ohodnocené grafy. Hamiltonovským cyklem rozumíme cyklus, tedy uzavřenou cestu v grafu obsahující každý vrchol maximálně jednou, který obsahuje všechny vrcholy. [16]

Pro snadnější pochopení problému uveďme krátký ukázkový příklad.

Příklad 6.3 (Ukázková úloha)**.** Mějme pět měst, kde některá jsou spojena silnicí, viz následující obrázek.



Obrázek 6.4: TSP: ukázkový příklad

Úkolem je nalézt nejkratší trasu, navštěvující všech pět měst, tak abychom každým městem projeli právě jednou a skončili ve městě, ve kterém jsme začali.

Nejprve převedeme grafické zadání úlohy do matice vah, která reprezentuje vzdálenosti jednotlivých vrcholů. Vrcholy, jenž nejsou spojeny cestou, mají mezi sebou vzdálenost ∞.

	A	**B**	**C**	**D**	**E**
A	0	10	15	8	∞
B	10	0	10	14	∞
C	15	10	0	12	7
D	8	14	12	0	20
E	∞	∞	7	20	0

(a) Matice vah

Posloupnost vrcholů	Délka trasy
(A, D, E, C, B)	55km
(A, B, C, E, D)	55km
(A, C, E, D, B)	66km
(A, B, D, E, C)	66km

(b) Přípustné cykly

Tabulka 6.5: TSP ukázkový příklad

Chceme-li nalézt nejkratší trasu požadovaných vlastností, můžeme zkoumat jednotlivé permutace vrcholů. Těchto permutací je však $5! = 120$. Jelikož se však jedná o cykly, nezáleží ve kterém vrcholu začneme, můžeme tedy brát vrchol 1 jako fixní začátek. Potom počet všech možných cyklů v grafu bude $4! = 24$. Výběr cyklů, které používají pouze

existující cesty (nemají nekonečnou délku), lze vidět v tabulce 6.5b, lze tedy vidět, že existují pouze dva různé cykly požadovaných vlastností, neboť v neorientovaném grafu lze zapsat cyklus dvěma směry.

Nejkratší cesta navštěvující každé město právě jednou a končící ve městě, kde začala, je dána posloupností měst (A, D, E, C, B) a má délku 55km. △

Poznámka 6.4. Pokud bychom chtěli najít nejkratší trasu bez požadavku na neopakování vrcholů, může být řešení jiné. Například ve výše uvedené ukázkové úloze je nejkratší trasa obsahující všechna města se shodným začátkem a koncem (A, D, C, E, C, B) o délce 54km.

Tuto upravenou úlohu lze převést na klasický TSP pomocí úpravy matice vah, tak aby váhy odpovídaly vždy nejkratší cestě mezi danými vrcholy. Tohoto lze dosáhnout například aplikací Dijkstrova algoritmu. [16]

6.2.1 Řešení TSP pomocí CE metody

V této části ukážeme použití CE metody pro nalezení řešení TSP, čerpáme především z [1]. Jako vstupní data problému budeme uvažovat úplný neorientovaný kladně hranově ohodnocený graf $G = (V, C)$ s kladnou symetrickou maticí vah jednotlivých hran $C = (c_{ij})$.

Jako výkonnostní funkci uvažujeme délku cyklu $x_1 \to \dots x_n \to x_1$ tedy

$$H(\boldsymbol{x}) = \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, x_1},\ \boldsymbol{x} = (x_1, \dots, x_n).$$

Oproti předchozím úlohám budeme nyní řešit minimalizační problém, tedy hledáme

$$\gamma^* = \min_{\boldsymbol{x} \in \mathcal{X}} H(\boldsymbol{x}),$$

kde množinou přípustných stavů budeme rozumět všechny Hamiltonovské cykly v grafu G. Pro účely simulace Hamiltonovský cyklů budeme používat ekvivalentní vyjádření těchto cyklů pomocí posloupností vrcholů tvořených permutacemi všech vrcholů množiny V. Je zřejmé, že se jeden Hamiltonovský cyklus dá vyjádřit pomocí více těchto posloupností (posunování počátečního vrcholu, např $(v_1, v_2, v_3) = (v_3, v_1, v_2)$), proto můžeme zafixovat první člen posloupnosti na libovolný vrchol.

Generování hodnot z množiny $\mathcal{X}$ můžeme realizovat pomocí Markovova řetězce s nahrazováním (MCWR), který byl již použit v úloze 5.11 a odvozen v sekci 5.3. V případě zafixování prvního vrcholu jako prvního prvku posloupnosti bude počáteční matice přechodu o velikosti $n \times n$ nabývat tvaru

$$\widetilde{\boldsymbol{P}}_0 = \begin{pmatrix} 0 & x & \cdots & x \\ \vdots & \vdots & & \vdots \\ 0 & x & \cdots & x \end{pmatrix}, x = \frac{1}{n-1}.$$

A formule pro aktualizaci parametrů bude

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \leq \gamma_t\right) \cdot I\left(\boldsymbol{X}_k \in \mathcal{X}_{ij}\right)}{\sum_{k=1}^{N} I\left(H\left(\boldsymbol{X}_k\right) \leq \gamma_t\right)},$$

kde $\mathcal{X}_{ij}$ značí množinu posloupností obsahující přechod z i-tého do j-tého vrcholu.

V případě minimalizačních úloh bude $\gamma_t = H_{\lceil \rho \cdot N \rceil}$, případně N_b bude určovat počet snímku s nejmenší výkonnostní funkcí.

Syntetická úloha Stejně jako v případě max-cut problému, musíme pro ověření efektivity použít uměle sestrojenou úlohu se známým řešením.

Jako jedno z řešení syntetického TSP problému o n vrcholech budeme uvažovat posloupnost $v_1 \to v_2 \to \ldots \to v_n$ s délkou minimální trasy $\gamma^* = n$. Váhy cest mezi jednotlivými vrcholy optimální posloupnosti $c_{i,i+1}, c_{i+1,i}$ pro $i = 1, \ldots, n-1$ a $c_{1,n}, c_{n,1}$ nastavíme na 1, zbylé váhy doplníme náhodnými čísly z $U\left(1, 10\right)$ tak, aby byla matice vah symetrická. Dostaneme tedy matici vah následujícího tvaru

$$C = \begin{pmatrix} 0 & 1 & c_{1,3} & \cdots & c_{1,n-1} & 1 \\ 1 & 0 & 1 & c_{2,4} & \cdots & c_{2,n} \\ c_{1,3} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & c_{n-2,n} \\ c_{1,n-1} & \cdots & c_{n-3,n-1} & 1 & 0 & 1 \\ 1 & c_{2,n} & \cdots & c_{n-2,n} & 1 & 0 \end{pmatrix}, c_{i,j} \in (1, 10)\,.$$

Jako testovací úlohu zvolíme počet vrcholů $n = 20$. Použijeme CE a FACE algoritmus s nastavením parametrů

- FACE: $\alpha = 0.5, d = 5, c = 3, \rho = 0.05$, $N_{min} = n^2, N_{MAX} = 20 \cdot N_{min}, \beta = N_{min}, N_b = \left\lceil \rho \cdot n^2 \right\rceil$,
- CE: $\alpha = 0.5, d = 5, \rho = 0.05, N = n^2 \cdot 5, N_b = \left\lceil \rho \cdot n^2 \right\rceil$.

Optimální délka trasy je 20, výsledky testování pro 100 běhů algoritmů lze nalézt v následující tabulce, sledovanými hodnotami je průměr času výpočtu $\overline{čas}$, maximální čas výpočtu $\max čas$, průměrný celkový počet použitých pokusů $\overline{N_{sum}}$, maximální celkový počet použitých pokusů $\max N_{sum}$, průměrný počet iterací $\overline{Iter}$, maximální počet iterací $\max Iter$, průměrný nalezený minimální výkon $\overline{H^*}$ a maximum z nalezených minimálních výkonů $\max H^*$.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\max H^*$
CE	0.15	0.17	23920	28000	11.96	14	20.1	23.6
FACE	0.25	0.55	23980	53600	17.25	23	20.8	24.58

Tabulka 6.6: Výsledky řešení syntetické TSP pro 20 měst (100 běhů)

Z těchto 100 běhů CE metoda nalezla řešení 97 krát a FACE metoda nalezla řešení 78 krát.

Otestujeme ještě řešení dvou velikostí úloh s $n = 40$ a $n = 80$. Úlohu s $n = 40$ opět řešíme 100 krát každým algoritmem, úlohu s $n = 80$ z časových důvodů pouze 10 krát.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\max H^*$
CE	7.83	9.18	186000	216000	23.25	27	40.75	42.64
FACE	2.28	5.55	100992	249600	36.1	46	41.63	45.04

Tabulka 6.7: Výsledky řešení syntetické TSP pro 40 měst (100 běhů)

Ze 100 běhů CE metoda nalezla řešení 66 krát a FACE metoda nalezla řešení pouze 52 krát.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\max H^*$
CE	309	353	1593600	1856000	49.8	58	82.83	85.87
FACE	156	235	843520	1286400	93.2	99	85.61	88.55

Tabulka 6.8: Výsledky řešení syntetické TSP pro 80 měst (10 běhů)

V případě úlohy s $n = 80$ nenalezla žádná z metod v 10 bězích optimální řešení.

Jak lze vidět z výsledků testování, CE metoda je účinným nástrojem pro řešení TSP. V případě dimenze úlohy $n = 40$ nalezla optimální řešení s 66% úspěšností, přičemž množina možných posloupností, mezi kterými se řešení nachází (pouze 2 řešení) má velikost $39! = 2.0398 \cdot 10^{46}$. FACE metoda dosahuje horších výsledků, avšak pro rozsáhlejší úlohy je značně rychlejší a vyžaduje méně pokusů. Dále lze vidět, že rostoucí dimenze úlohy značně navyšuje výpočetní čas, což je motivací pro paralelní implementaci, viz sekce 8.3.2.

6.2.2 TSP s body v rovině

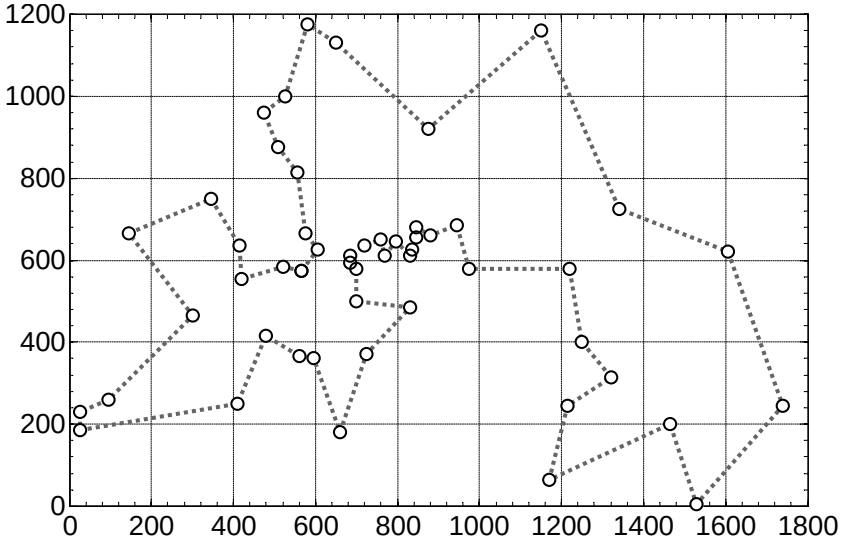
V této části budeme řešit TSP s body v rovině, kdy je matice vah dána euklidovskou vzdáleností jednotlivých bodů. Tyto úlohy splňují trojúhelníkovou nerovnost, tedy

$$c_{i,j} + c_{j,k} \geq c_{i,k},$$

což nemuselo v předchozích úlohách platit.

Pro testování volíme úlohu *berlin52.tsp* z knihovny TSPLIB[5] s 52 místy v Berlíně. Optimální řešení této úlohy má délku 7544 a lze vidět na obrázku 6.5.

[5]Dostupné na http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.



Obrázek 6.5: TSP s body v rovině, 52 měst

Pro řešení této úlohy bude použita GPU implementace a dvě verze CE metody:

CE-1: původní CE metoda s parametry $\alpha = 0.5, d = 5, \rho = 0.05, N = 10 \cdot n^2 \cdot, N_b = \lceil \rho \cdot n^2 \rceil$.

CE-2: stejné parametry jako v případě CE-1, avšak α se volí dle vzorce 3.10 s parametry $\beta = 0.9, q = 15$ a používá se modifikovaná počáteční matice přechodu. Modifikace spočívá ve využití matice vah C:

$$\widetilde{\boldsymbol{P}}_0 = (1-\epsilon)\cdot \boldsymbol{P}_0 + \epsilon \cdot \boldsymbol{g},\ \boldsymbol{g} = (g_{ij}) = \begin{cases} \frac{1}{c_{ij}^5} / \sum_{k=1,k\neq i}^{n} \frac{1}{c_{i,k}^5}, & i \neq j \\ 0, & i = j \end{cases},$$

kde $\boldsymbol{P}_0$ je počáteční matice přechodu nepreferující žádnou trasu (jako v případě CE-1) a $\epsilon = 0.9$.

Výsledky lze vidět v následující tabulce.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	max H^*
CE-1	1.71	2.01	999475	1216512	36.2	44	7760	8151
CE-2	0.76	0.89	444658	540800	16.4	20	7581	7703

Tabulka 6.9: Výsledky TSP s body rovině, 52 měst (10 běhů)

Z výsledků testování lze vidět, že případné modifikace CE metody s ohledem na charakter problému mohou značně zlepšit konvergenci metody. Průměrná relativní chyba klasické CE metody je $\varepsilon = \frac{7760-7544}{7544} = 0.0286$, přičemž při modifikaci metody s ohledem na řešenou úlohu je průměrná relativní chyba $\varepsilon = \frac{7581-7544}{7544} = 0.0049$, což je 5.8 krát vyšší relativní přesnost. Výrazné zlepšení také nastalo v celkovém počtu požitých náhodných pokusů, které je oproti nemodifikované CE metodě méně než poloviční.

6.3 Problém zarovnání sekvencí

Tato sekce čerpá převážně z [11, 21, 1]. S problémem zarovnání sekvencí („*sequence alignment*") se často setkáváme v počítačové biologii. Jedná se o úlohu na hledání optimálního zarovnání dvou či více sekvencí. Pojmy sekvence a zarovnání objasní následující definice.

Definice 6.5 (Sekvence)**.** Nechť je dána konečná diskrétní množina Σ tvořící abecedu prvků, pak posloupnosti $\boldsymbol{s} = s_1, \ldots, s_n, s_i \in \Sigma$ říkáme *sekvence* o délce n.

Definice 6.6 (Zarovnání)**.** Nechť $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_p), p \geq 2$ je množina sekvencí o délkách n_i nad abecedou Σ. Zarovnáním p sekvencí rozumíme množinu $A = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_p)$ sekvencí délky $k \geq \max n_i$, kde $\boldsymbol{a}_i$ je sekvence délky $k \geq n_i$ nad $\Sigma \cup \{-\}$, která vznikla ze sekvence $\boldsymbol{s}_i$ vložením $k - n_i$ mezer " $-$ ".

V dalším textu se budeme zabývat pouze zarovnáním dvou sekvencí. Optimálním zarovnáním rozumíme zarovnání dosahující minimální hodnoty dané výkonnostní funkce

$$\min_{A \in \mathcal{X}} H(A),$$

kde množinou přípustných stavů $\mathcal{X}$ rozumíme všechna zarovnání, kde nenastane $(\boldsymbol{a}_1)_i = (\boldsymbol{a}_2)_i =$ " $-$ ", tedy nezarovnáváme dvě mezery na stejném místě. Výkonnostní funkce může být definována například počtem rozdílných hodnot v zarovnání daných sekvencí

$$H(A) = \sum_{i=1}^{k} I\left((\boldsymbol{a}_1)_i \neq (\boldsymbol{a}_2)_i\right). \tag{6.1}$$

Obecně však může být výkonnostní funkce mnohem složitější, my však budeme dále používat pouze výkonnostní funkci danou vzorcem 6.1. V případech, kdy výkonnostní funkce závisí na všech prvcích jednotlivých posloupností, je problém zarovnání posloupností NP-kompletním problém. [1, str. 230]

Pro odvození aplikace CE metody na tento problém začněme s krátkým ukázkovým příkladem.

Příklad 6.7. Mějme dvě posloupnosti $\boldsymbol{s}_1 = ACTGGA$ a $\boldsymbol{s}_2 = AGTGCAGATA$ nad abecedou $\Sigma = \{A, C, G, T\}$. Úkolem je najít optimální zarovnání vzhledem k výkonnostní funkci 6.1. Dvě různá zarovnání těchto sekvencí lze vidět v tabulce níže.

ACTG--GA--	-ACTGG--A---
AGTGCAGATA	AG-TGCAG-ATA
(a) $A_1 : H(A_1) = 5$	(b) $A_2 : H(A_2) = 10$

Tabulka 6.10: Dvě různá zarovnání sekvencí

Je zřejmé, že minimální výkonnost, které lze dosáhnout zarovnáním dvou sekvencí o délkách n_1,n_2 je větší nebo rovna $|n_1 - n_2|$. V tomto případě je minimální výkonost 5 dosažena zarovnáním A_1 (důvodem je umístnění znaku C v obou sekvencích). $\triangle$

Všimněme si, že při zarovnání dvou sekvencí, může v každém členu zarovnání $\{(\boldsymbol{a}_1)_i, (\boldsymbol{a}_2)_i\}$ nastat jedna ze tří možností:

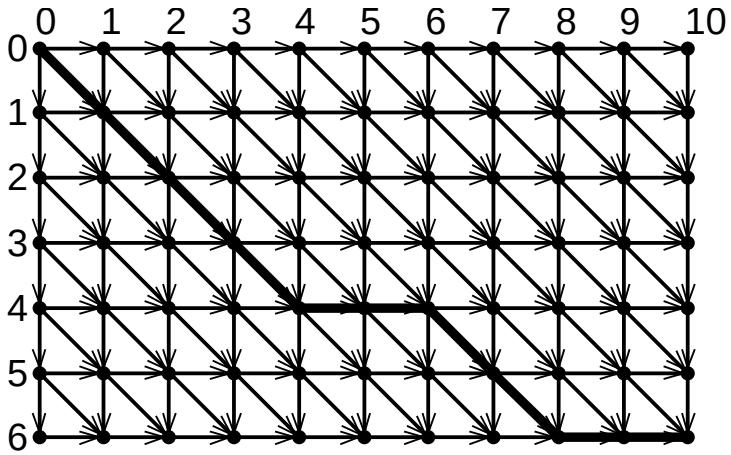
1. $(\boldsymbol{a}_1)_i$ je další prvek sekvence $\boldsymbol{s}_1$ a $(\boldsymbol{a}_2)_i$ je mezera, nebo
2. $(\boldsymbol{a}_1)_i$ je mezera a $(\boldsymbol{a}_2)_i$ je další prvek sekvence $\boldsymbol{s}_2$, nebo
3. $(\boldsymbol{a}_1)_i$ je další prvek sekvence $\boldsymbol{s}_1$ a $(\boldsymbol{a}_2)_i$ je další prvek sekvence $\boldsymbol{s}_2$.

Tedy zarovnání A lze ekvivalentně přeformulovat jako posloupnost těchto tří možností $\boldsymbol{x} = (x_1, \ldots, x_k), x_i \in \{1, 2, 3\}$. Je zřejmé, že platí-li

$$\sum_{i=1}^{k} I(x_i = 1) + \sum_{i=1}^{k} I(x_i = 3) = n_1 \wedge \sum_{i=1}^{k} I(x_i = 2) + \sum_{i=1}^{k} I(x_i = 3) = n_2,$$

jedná se o zarovnání z množiny přípustných stavů $\mathcal{X}$, a množina všech vektorů $\boldsymbol{x} : x_i \in \{1, 2, 3\}$ splňující tuto podmínku tvoří celou množinu přípustných stavů $\mathcal{X}$. Těmto vektorům budeme říkat vektory zarovnání (*„alignment vector"*).

Dále lze vektory zarovnání reprezentovat pomocí cest zarovnání (*„alignment path"*) v grafu zarovnání (*„alignment graph"*), viz následující obrázek reprezentující zarovnání A_1.



Obrázek 6.6: Reprezentace zarovnání dvou sekvencí jako cesty v grafu

Graf zarovnání je orientovaný graf tvořen $(n_1 + 1) \times (n_2 + 1)$ vrcholy zarovnanými do mřížky (indexovat budeme vrcholy (i, j), kde i je index řádku a j index sloupce) s třemi typy hran odpovídajícími předchozím třem typům prvků vektoru zarovnání:

1. hrana vedoucí z každého vrcholu (mimo poslední řádek mřížky) dolů,

2. hrana vedoucí z každého vrcholu (mimo poslední sloupec mřížky) doprava a
3. hrana vedoucí z každého vrcholu (mimo poslední řádek a sloupec mřížky) po diagonále doprava dolů.

Cestou zarovnání rozumíme libovolnou cestu v orientovaném grafu zarovnání s počátečním vrcholem $(0, 0)$ a koncovým vrcholem (n_1, n_2). Získali jsme tedy další ekvivalentní formulaci množiny přípustných zarovnání $\mathcal{X}$ a to množinu všech cest zarovnání.

Tato poslední formulace nám poskytuje návod, jak generovat jednotlivé zarovnání dvou sekvencí.

6.3.1 Řešení zarovnání dvou sekvencí pomocí CE metody

V předchozí části jsme ukázali ekvivalentní formulaci zarovnání dvou sekvencí pomocí cesty zarovnání, tu nyní použijeme ke generaci jednotlivých zarovnání sekvencí. Jak už bylo zmíněno v sekci 5.3, cesty v grafu lze efektivně simulovat pomocí Markovových řetězců.

Uvažovaný MŘ bude mít následující vlastnosti:

- počet stavů bude roven počtu vrcholů grafu zarovnání,
- nenulové pravděpodobnosti přechodu budou jen tam, kde se v grafu zarovnání nachází hrana,
- stav odpovídající vrcholu (n_1, n_2) je absorpční stav a
- počáteční stav je stav odpovídající vrcholu $(0, 0)$.

Dále budeme uvažovat posloupnost stavů MŘ odpovídající této posloupnosti vrcholů:

$$(0,0), (1,0), \ldots, (n_1,0), (0,1), \ldots, (n_1-1, n_2), (n_1 n_2).$$

Matici přechodu odpovídající MŘ reprezentující úlohu na zarovnání dvou sekvencí z příkladu 6.7, která nepreferuje žádný přechod (vhodná jako počáteční matice parametrů pro CE úlohu), lze vidět na obrázku E.1. Z konstrukce této matice přechodu je zřejmé, že v každém řádku matice jsou maximálně tři nenulové hodnoty, což vyplývá z tvaru vektoru zarovnání $\boldsymbol{x} \in \{1, 2, 3\}^k$. Z tohoto důvodu budeme uchovávat pouze pravděpodobnosti pro přechod do stavů reprezentující jeden z vrcholů $(i+1, j), (i, j+1)$ a $(i+1, j+1)$. Implementačně lze toto v Matlabu řešit pomocí trojrozměrné matice o velikostech $(n_1 + 1) \times (n_2 + 1) \times 3$, kde

- prvky $(i, j, 1)$ označují přechod do vrcholu $(i+1, j)$,
- prvky $(i, j, 2)$ označují přechod do vrcholu $(i, j+1)$ a
- prvky $(i, j, 3)$ označují přechod do vrcholu $(i+1, j+1)$.

Tato reprezentace nebude odpovídat pro prvky matice přechodu odpovídající stavu pro vrchol (n_1, n_2), tento stav je však absorpční a při jeho dosažení MŘ ukončujeme. Tuto reprezentaci matice přechodu pro úlohu 6.7 lze vidět na obrázku E.2. Počáteční matici $\widetilde{\boldsymbol{P}}_0$ pro CE metodu řešící zarovnání dvou posloupností o délkách n_1 a n_2 budeme konstruovat následovně:

$$\widetilde{p}_{0,ij1} = \begin{cases} 1/3, & i \in \{0,\ldots,n_1-1\}, j \in \{0,\ldots,n_2-1\} \\ 1, & i \in \{0,\ldots,n_1-1\}, j = n_2 \\ 0, & i = n_1, j \in \{0,\ldots,n_2\} \end{cases},$$

$$\widetilde{p}_{0,ij2} = \begin{cases} 1/3, & i \in \{0,\ldots,n_1-1\}, j \in \{0,\ldots,n_2-1\} \\ 1, & i = n_1, j \in \{0,\ldots,n_2-1\} \\ 0, & i \in \{0,\ldots,n_1\}, j = n_2 \end{cases},$$

$$\widetilde{p}_{0,ij3} = \begin{cases} 1/3, & i \in \{0,\ldots,n_1-1\}, j \in \{0,\ldots,n_2-1\} \\ 0, & i \in \{0,\ldots,n_1-1\}, j = n_2 \\ 0, & i = n_1, j \in \{0,\ldots,n_2\} \end{cases}.$$

Získali jsme tedy postup pro generaci zarovnání sekvencí. Dále potřebujeme formule pro aktualizaci matice přechodu, ty však získáme jednoduše, neboť se jedná o MŘ:

$$\widetilde{p}_{t,ijm} = \frac{\sum_{l=1}^{N} I\left(H\left(\boldsymbol{X}_l\right) \leq \gamma_t\right) \cdot I\left(\boldsymbol{X}_l \in \mathcal{X}_m\left(i,j\right)\right)}{\sum_{l=1}^{N} I\left(H\left(\boldsymbol{X}_l\right) \leq \gamma_t\right) \cdot I\left(\boldsymbol{X}_l \in \mathcal{X}\left(i,j\right)\right)},$$

kde $\boldsymbol{X_l}$ značí l-tou cestu zarovnání, $m \in \{1,2,3\}, i \in \{0,\ldots,n_1\}, j \in \{0,\ldots,n_2\}, \mathcal{X}_1\left(i,j\right)$ značí množinu všech cest s přechodem $(i,j) \to (i+1,j)$, $\mathcal{X}_2\left(i,j\right)$ značí množinu všech cest s přechodem $(i,j) \to (i,j+1)$, $\mathcal{X}_3\left(i,j\right)$ značí množinu všech cest s přechodem $(i,j) \to (i+1,j+1)$ a $\mathcal{X}\left(i,j\right)$ značí množinu všech cest, které navštívily vrchol (i,j).

Jako výkonnostní funkci budeme používat

$$\overline{H}\left(\boldsymbol{X}\right) = H\left(A\right) = \sum_{i=1}^{k} I\left(\left(\boldsymbol{a}_1\right)_i \neq \left(\boldsymbol{a}_2\right)_i\right), \boldsymbol{X} \Leftrightarrow A,$$

čímž jsme získali vše potřebné pro aplikaci CE metody na danou úlohu.

6.3.2 Zarovnání proteinových řetězců

V této části použijeme CE metodu pro zarovnání proteinových sekvencí, jelikož se jedná o rozsáhlejší úlohy, použijeme GPU implementaci odvozenou v sekci 8.3.3. Data proteinových řetězců včetně popisu jejich funkce byla získána ze serveru UniProt viz [22]. Pro demonstraci CE metody použijeme dvě úlohy.

Příklad 6.8 (Zarovnání proteinových řetězců: Escherichia coli: Nitrogen Regulatory protein P-II 1 (P0A9Z1) a Nitrogen Regulatory protein P-II 1 (P0AC55))**.**

V této úloze budeme zarovnávat dva proteinové řetězce bakterie Escherichia coli, které nepřímo ovlivňují přepis genu pro syntézu glutaminu. Vstupními proteinovými řetězci jsou sekvence:

s_1 = MKKIDAIIKPFKLDDVREALAEVGITGMTVTEVKGFGRQKGHTELYRGAEYMVDFLPKVKIEIVV
PDDIVDTCVDTIIRTAQTGKIGDGKIFVFDVARVIRIRTGEEDDAAI,

s_2 = MKLVTVIIKPFKLEDVREALSSIGIQGLTVTEVKGFGRQKGHAELYRGAEYSVNFLPKVKIDVAI
ADDQLDEVIDIVSKAAYTGKIGDGKIFVAELQRVIRIRTGEADEAAL,

o délce $n_1 = 112$ a $n_2 = 112$.

Pro řešení použijeme standardní CE metodu s nastavením parametrů: $\alpha = 0.9, d = 5, \rho = 0.05, N = n_1 \cdot n_2, N_b = \lceil \rho \cdot n_1 \cdot n_2 \rceil$. Výsledky pro 100 běhů lze vidět v tabulce 6.11 a jedno z optimálních zarovnání lze vidět v tabulce 6.12.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	max H^*
CE	0.15	0.17	166260	177408	13.12	14	36	36

Tabulka 6.11: Výsledky řešení zarovnání proteinových řetězců (100 běhů)

MKKIDAIIKPFKLDDVREALAEVGITGMTVTEVKGFGRQKGHTELYRGAEYMVDFLPKVKIE
MKLVTVIIKPFKLEDVREALSSIGIQGLTVTEVKGFGRQKGHAELYRGAEYSVNFLPKVKID
IVVPDDIVDTCVDTIIRTA-QTGKIGDGKIFVFDVARVIRIRTGEEDDAAI
VAIADDQLDE-VIDIVSKAAYTGKIGDGKIFVAELQRVIRIRTGEADEAAL

Tabulka 6.12: Výsledné zarovnání sekvencí s_1 a s_2 úlohy 6.8

Z výsledků lze vidět, že CE metoda nalezla ve všech běžích řešení s optimální výkonností 36. △

Příklad 6.9 (Zarovnání proteinových řetězců: Brugia malayi Vab-3 protein (A8PZ80) a Loa loa Vab-3 protein (E1FTG0)).

V této úloze budeme zarovnávat dva proteinové řetězce řídící vývoj zrakových receptorů parazitů Brugia malayi a Loa loa (Vlasovec oční), oba tyto paraziti způsobují nemoc zvanou Filarióza. Vstupními proteinovými řetězci jsou sekvence:

s_1 = MKLIVDSGHTGVNQLGGVFVNGRPLPDSTRQKIVDLAHQGARPCDISRILQVSNGCVSKILCRYY
ESGTIRPRAIGGSKPRVATVSVCDKIESYKREQPSIFAWEIRDKLLHEKVCSPDTIPSVSSINRV
LRNLAAKKEQQAMQNDFYDRALRYSSTQWYNQWPMGVPSAVGLAQLPPLTQANHLNKKDSGKLLW
YSNYKGGWKELLI

s_2 = MLGDKKYSGHTGVNQLGGVFVNGRPLPDSTRQKIVDLAHQGARPCDISRILQVSNGCVSKILCRY
YESGTIRPRAIGGSKPRVATVSVCDKIESYKREQPSIFAWEIRDKLLHEKVCSPDTIPSIHVGFS
IHF

o délce $n_1 = 208$ a $n_2 = 133$.

Pro řešení použijeme standardní CE metodu s nastavením parametrů: $\alpha = 0.9, d = 5, \rho = 0.05, N = n_1 \cdot n_2, N_b = \lceil \rho \cdot n_1 \cdot n_2 \rceil$. Výsledky pro 100 běhů lze vidět v tabulce 6.13 a jedno z optimálních zarovnání lze vidět v tabulce 6.14.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	max H^*
CE	0.52	0.6	412704	460800	14.33	16	88,25	89

Tabulka 6.13: Výsledky řešení zarovnání DNA řetězců a (100 běhů)

MKLIV-D-SGHTGVNQLGGVFVNGRPLPDSTRQKIVDLAHQGARPCDISRILQVSNGCVSKI
M-LGDKKYSGHTGVNQLGGVFVNGRPLPDSTRQKIVDLAHQGARPCDISRILQVSNGCVSKI
LCRYYESGTIRPRAIGGSKPRVATVSVCDKIESYKREQPSIFAWEIRDKLLHEKVCSPDTIP
LCRYYESGTIRPRAIGGSKPRVATVSVCDKIESYKREQPSIFAWEIRDKLLHEKVCSPDTIP
SVSSINRVLRNLAAKKEQQAMQNDFYDRALRYSSTQWYNQWPMGVPSAVGLAQLPPLTQANH
S---IH-V------G-F----SIHF----------
LNKKDSGKLLWYSNYKGGWKELLI

Tabulka 6.14: Výsledné zarovnání sekvencí s_1 a s_2 úlohy 6.9

V případě této úlohy bylo v 75% běžích algoritmu nalezeno optimální řešení o výkonnosti 88. △

6.4 Srovnání CE metody s jinými simulačními metodami při řešení TSP

V této části srovnáme efektivitu CE metody pro optimalizace na TSP úloze s jinými dostupnými metodami pro řešení kombinatorických úloh. Jako další simulační metody pro řešení optimalizačních problémů volíme:

- genetické algoritmy,
- simulované žíhání a
- mravenčí kolonie.

Popis těchto metod je nad rámec této práce, více o optimalizaci pomocí genetických algoritmů lze najít v [8], optimalizaci pomocí simulovaného žíhání lze nalézt v [9] a optimalizaci pomocí mravenčích kolonií lze nalézt v [10].

Jednotlivé implementace algoritmů (mimo CE metodu) byly převzaty z webu *Matlab central-file exchange*:

- Genetický algoritmus: *http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm*
- Algoritmus simulovaného žíhání: *http://www.mathworks.com/matlabcentral/fileexchange/9612-traveling-salesman-problem–tsp–using-simulated-annealing*
- Algoritmus optimalizace mravenčí kolonií: *http://www.mathworks.com/matlabcentral/fileexchange/33448-ant-system-tsp-solver*

6.4.1 Testování

Jako testovací TSP úlohu volíme problém ze sekce 6.2.2 s $n = 52$ vrcholy.

Parametry jednotlivých metod volíme s ohledem na optimální výkon algoritmu, a pro jednotlivé algoritmy jsou:

genetický algoritmus: velikost populace 2000, počet iterací 2000.

simulované žíhání: počáteční teplota 200, ochlazovací faktor 0.9995, počet iterací 60000, počet modifikovaných vrcholů v iteraci 1.

mravenčí kolonie: počet iterací 20000, velikost kolonie 100, rychlost odpařování feromonu 0.1, váha feromonové cesty na řešení 1, váha heuristické modifikace na řešení 3.

Pro CE metodu testujeme dvě verze (viz sekce 6.2.2):

1. CE metodu bez modifikací, tedy počáteční matice přechodu nepreferuje žádné trasy a parametr α volíme konstantní.
2. CE metodu s modifikovanou počáteční maticí přechodu sestavenou s ohledem na matici vah jednotlivých cest s proměnným parametrem α dle vzorce 3.10.

Výsledky srovnání lze vidět v následující tabulce, sledovanými hodnotami je čas běhu (průměr a maximum) a nalezené řešení (průměr a maximum).

metoda	$\overline{čas}$	$\max čas$	$\overline{H^*}$	$\max H^*$	počet správných řešení
CE metoda	71.8	80.4	7760	8151	1
upravená CE metoda	49.0	59.0	7581	7703	5
Genetické algoritmy	36.8	37.4	7729	8091	5
Simulované žíhání	101	104.1	7980	8305	0
Mravenci	30.8	31.7	7647	7681	0

Tabulka 6.15: Výsledky řešení TSP pro 52 měst (10 běhů)

Jak lze vidět z výsledků testování, upravená CE metoda je nejpřesnější, má nejmenší průměr nalezených řešení a správné řešení nalezla v 50% běhů. Časy běhu jednotlivých metod jsou srovnatelné a slouží pouze jako orientační měřítko, neboť ačkoliv jsou všechny algoritmy testovány na stejném hardware, může mít jiná implementace stejné metody jinou dobu běhu.

Z provedeného srovnání je zřejmé, že je CE metoda užitečným nástrojem pro kombinatorické optimalizační úlohy srovnatelným a v některých případech lepším než jiné simulační metody řešení.

7 Cross-entropy metoda pro řešení spojitých optimalizací

V této části ukážeme využití CE metody pro řešení spojitých vícerozměrných optimalizačních úloh, obsahujících více lokálních extrémů. Oproti problémům na hledání lokálních extrémů, kde můžeme často využít efektivních gradientních nebo Newtonovských metod (viz [23]), zajímá nás nyní pouze jeden globální extrém. Tato část převážně čerpá z [1, kap. 5].

Formulace řešeného optimalizačního problému bude tedy

$$\min_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}), \text{ nebo } \max_{\boldsymbol{x}\in\mathcal{X}} H(\boldsymbol{x}),$$

kde $H(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ a $\mathcal{X} \subset \mathbb{R}^n$ množina na níž hledáme globální extrém.

Pro řešení spojitých optimalizačních problémů použijeme stejné schéma CE metody jako při kombinatorických optimalizacích. Je tedy třeba sestavit postup generování bodů v $\mathbb{R}^n$, výkonnostní funkci $H(\boldsymbol{x})$ a formuli pro aktualizaci parametrů.

Vhodnou volbou pro generování bodů v $\mathbb{R}^n$ je vícerozměrné normální rozdělení s nezávislými složkami $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. Toto rozdělení je vhodné především díky parametru středních hodnot $\boldsymbol{\mu}$, který určuje bod v jehož okolí je největší pravděpodobnost vygenerování $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. Jako počáteční parametry rozdělení budeme volit $\boldsymbol{\mu}, \boldsymbol{\sigma}$ tak, aby $\mathbb{P}(\boldsymbol{X} \in \mathcal{X})$ byla dostatečně veliká.

Jako výkonnostní funkci volíme například optimalizovanou funkci $H(\boldsymbol{x})$ s kombinací přípustné množiny $\mathcal{X}$:

$$\tilde{H}(\boldsymbol{x}) = H(\boldsymbol{x}) + \phi(\boldsymbol{x}),$$

kde $\phi(\boldsymbol{x})$ je penalizační funkce, například $\phi(\boldsymbol{x}) = \beta \cdot \operatorname{dist}(\boldsymbol{x}, \mathcal{X})$.

Pro aktualizaci parametru střední hodnoty $\boldsymbol{\mu}$ použijeme dříve odvozenou formuli (v případě hledání maxima)

$$\widetilde{\mu}_{t,j} = \frac{\sum_{i=1}^{N} I(H(\boldsymbol{X}_i) \geq \gamma_t) \cdot X_{ij}}{\sum_{i=1}^{N} I(H(\boldsymbol{X}_i) \geq \gamma_t)}.$$

Aktualizaci parametru rozptylu $\boldsymbol{\sigma}^2$ zde odvodíme. Nejprve spočteme

$$\begin{aligned}
\nabla_{\boldsymbol{\sigma}} \ln(f(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\sigma})) &= \nabla_{\boldsymbol{\sigma}} \ln\left(\prod_{i=1}^{n} \frac{1}{\sigma_i \cdot \sqrt{2\cdot\pi}} \cdot e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}\right) \\
&= \nabla_{\boldsymbol{\sigma}} - \sum_{i=1}^{n} \left[\frac{(x-\mu_i)^2}{2\sigma_i^2} + \ln\sigma_i + \ln\sqrt{2\cdot\pi}\right] \\
&= \left(\frac{(x-\mu_i)^2}{\sigma_i^3} - \frac{1}{\sigma_i}\right)_{i=1,\ldots,n} = \left(\frac{(x-\mu_i)^2 - \sigma_i^2}{\sigma_i^3}\right)_{i=1,\ldots,n},
\end{aligned}$$

což díky předem spočtené hodnotě $\tilde{\boldsymbol{\mu}}_t$ poskytuje explicitní formuli pro aktualizaci $\tilde{\boldsymbol{\sigma}}_t$:

$$\widetilde{\sigma}^2_{t,j} = \frac{\sum_{i=1}^{N} I(H(\boldsymbol{X}_i) \geq \gamma_t) \cdot (X_{ij} - \widetilde{\mu}_{t,j})^2}{\sum_{i=1}^{N} I(H(\boldsymbol{X}_i) \geq \gamma_t)}.$$

Navržený postup vyzkoušíme na krátké ukázkové úloze.

Příklad 7.1. Mějme funkci

$$h(x) = 1 - \cos(20 \cdot (x-2)) \cdot e^{-(x-2)^2},$$

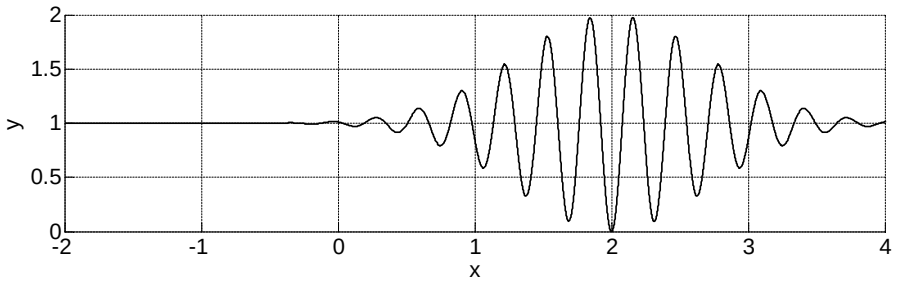
pro kterou chceme zjistit globální minimum na intervalu $\langle -3, 3 \rangle$. Graf této funkce lze vidět na obrázku 7.1a. Z konstrukce a grafu funkce je zřejmé, že globální minimum je dosaženo v bodě $x^* = 2$ a nabývá hodnoty $h(x^*) = 0$.

Pro řešení použijeme standardní CE metodu s nastavením parametrů $N = 100, N_b = 10$ a $\alpha = 0.9$, jako ukončovací podmínku volíme

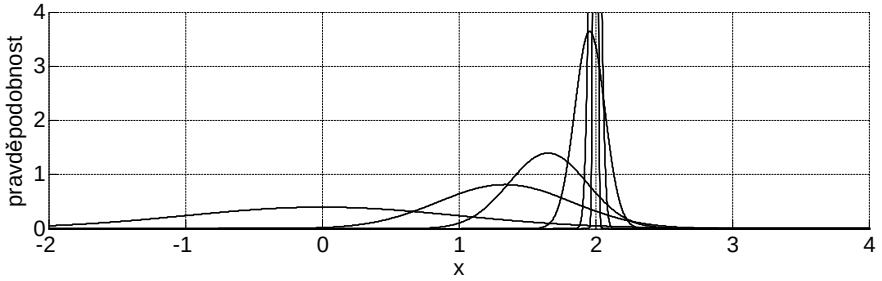
$$\max_{j \in 1,\ldots,n} (\widetilde{\sigma}_{t,j}) \leq \varepsilon$$

pro $\varepsilon = 10^{-4}$. Počáteční parametry hustoty pravděpodobnosti volíme $\widetilde{\mu}_0 = 0, \widetilde{\sigma}_0 = 1$.

Řešení bylo dosaženo v 8 iteracích, přičemž vzdálenost nalezeného výsledku od skutečného řešení byla $|x^* - \hat{x}| = 1.13 \cdot 10^{-6}$. Vývoj hustot pravděpodobnosti CE metody lze vidět na obrázku 7.1b.



(a) Optimalizovaná funkce



(b) Vývoj hustot pravděpodobnosti (1.-6. iterace)

Obrázek 7.1: Spojité optimalizace: ukázkový příklad

△

7.1 Benchmarkové úlohy a testování

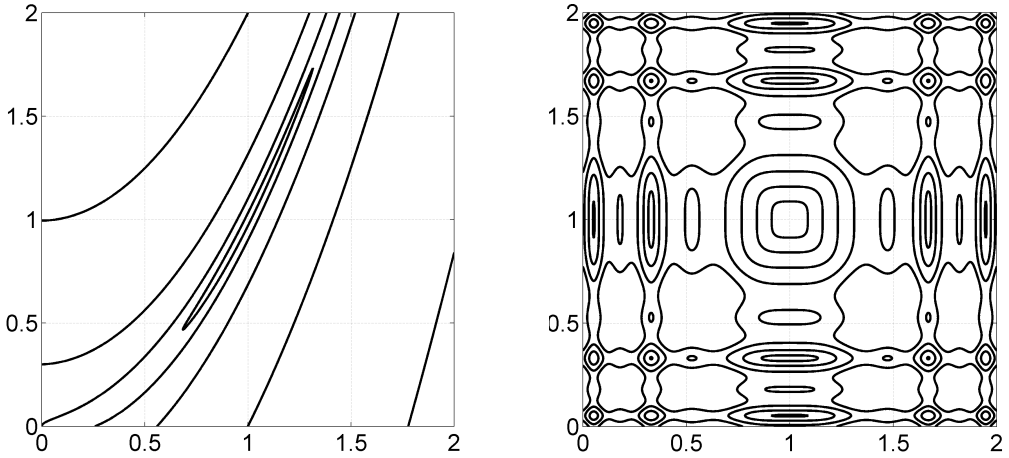
Jako benchmarkové úlohy pro otestování CE metody volíme následující funkce: *Rosenbrokovu* funkci

$$H(\boldsymbol{x}) = \sum_{i=1}^{n-1} 100 \cdot \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \tag{7.1}$$

a *trigonometrickou* funkci

$$H(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} 8 \cdot \sin^2\left(\eta \cdot (x_i - x_i^*)^2\right) + 6 \cdot \sin^2\left(2 \cdot \eta \cdot (x_i - x_i^*)^2\right) + \xi \cdot (x_i - x_i^*)^2 . \tag{7.2}$$

Globální minimum *Rosenbrockovy* funkce je v bodě $\boldsymbol{x}^* = (1, 1, \ldots, 1)$ o hodnotě $H(\boldsymbol{x}^*) = 0$ a globální minimum *trigonometrické* funkce je ve zvoleném bodě $\boldsymbol{x}^*$ o hodnotě $H(\boldsymbol{x}^*) = 1$. Ukázky vrstevnic *Rosenbrockovy* funkce a *trigonometrické* funkce pro $\eta = 7, \xi = 1, \boldsymbol{x}^* = (1, 1)$ lze vidět na obrázcích 7.2a a 7.2b.



(a) $H(\boldsymbol{x}) = (0.1, 1, 10, 100, 1000)$ (b) $H(\boldsymbol{x}) = (1.1, 2, 5, 10, 15)$

Obrázek 7.2: Benchmarkové úlohy v $\mathbb{R}^2$ (zleva: *Rosenbrockova* f., *trigonometrická* f.)

Poznámka 7.2. Vzhledem k výsledkům numerických testů bylo provedeno několik úprav CE algoritmu:

- pro $t > 1$ iteraci je do výběru $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim \mathcal{N}(\widetilde{\boldsymbol{\mu}}_t, \widetilde{\boldsymbol{\sigma}}_t)$ zařazen snímek $\boldsymbol{X}^*$ s doposud nejlepší (minimální, nebo maximální) hodnotou $H(\boldsymbol{X})$,
- vyhlazovací parametr α je volen zvlášť pro aktualizaci $\boldsymbol{\mu}$ a $\boldsymbol{\sigma}$, neboť pro aktualizaci $\boldsymbol{\sigma}$ je vzhledem k numerické stabilitě nutné použít mnohem menší vyhlazovací parametr,
- oceňovací funkce („*reward function*") $S(\boldsymbol{x}) = I(H(\boldsymbol{x}) \geq \gamma)$ může být změněna na tzv. váženou variantu:

$$S(\boldsymbol{x}) = I(H(\boldsymbol{x}) \geq \gamma) \cdot \psi(H(\boldsymbol{x})),$$

kde $\psi\left(H\left(\boldsymbol{x}\right)\right)$ udává váhu daného snímku. Například v pro úlohu $\max_{\boldsymbol{x}\in\mathcal{X}} H\left(\boldsymbol{x}\right)$, za předpokladu $H\left(\boldsymbol{x}\right) \geq 0\ \forall \boldsymbol{x} \in \mathcal{X}$, můžeme volit

$$\psi\left(H\left(\boldsymbol{x}\right)\right) = H\left(\boldsymbol{x}\right),$$

nebo pro úlohu $\min_{\boldsymbol{x}\in\mathcal{X}} H\left(\boldsymbol{x}\right)$, za předpokladu $H\left(\boldsymbol{x}\right) \geq 0\ \forall \boldsymbol{x} \in \mathcal{X}$, můžeme volit

$$\psi\left(H\left(\boldsymbol{x}\right)\right) = e^{-H(\boldsymbol{x})/\beta} \text{ případně } \psi\left(H\left(\boldsymbol{x}\right)\right) = 1/\left(\beta + H\left(\boldsymbol{x}\right)\right).$$

V případě použití vážené varianty oceňovací funkce přejdou aktualizační formule parametrů na

$$\widetilde{\mu}_{t,j} = \frac{\sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \gamma_t\right) \cdot X_{ij} \cdot \psi\left(H\left(\boldsymbol{X}_i\right)\right)}{\sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \gamma_t\right) \cdot \psi\left(H\left(\boldsymbol{X}_i\right)\right)},$$

$$\widetilde{\sigma}^2_{t,j} = \frac{\sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \gamma_t\right) \cdot \left(X_{ij} - \widetilde{\mu}_{t,j}\right)^2 \cdot \psi\left(H\left(\boldsymbol{X}_i\right)\right)}{\sum_{i=1}^{N} I\left(H\left(\boldsymbol{X}_i\right) \geq \gamma_t\right) \cdot \psi\left(H\left(\boldsymbol{X}_i\right)\right)}.$$

Jednotlivé benchmarkové funkce otestujeme pro $\mathcal{X} = \langle -2, 2\rangle^n$ a dimenze optimalizačního problému $n = 2, ., 10$, v případě *trigonometrické* funkce volíme nastavení $\eta = 7, \xi = 1, \boldsymbol{x}^* = (1, \ldots, 1)$.

Budeme testovat CE metodu, FACE metodu a Matlab optimization toolbox s následujícím nastavením

- CE: $N = 1000, N_b = 20, \alpha_\mu = 0.8, \alpha_\sigma = 0.9 - 0.9 \cdot (1 - 1/t)^6$,
- FACE: $N_{min} = 100, N_{max} = 10 \cdot N_{min}, N_b = 20, \alpha_\mu = 0.8, \alpha_\sigma = 0.9 - 0.9 \cdot (1 - 1/t)^6$,
- Matlab optimization toolbox: Algorithm: interior-point, MaxFunEvals: 10^7, MaxIter: 10^4, TolCon: 10^{-6}, TolFun: 10^{-6}, TolX: 10^{-10}, FinDiffRelStep: $1.49 \cdot 10^{-8}$.

Při použití CE a FACE metody volíme penalizační funkci

$$\phi\left(\boldsymbol{x}\right) = 100 \cdot \sqrt{\sum_{i=1}^{n} \min\left\{x_i + 2, 0\right\}^2 + \min\left\{2 - x_i, 0\right\}^2}$$

a jako počáteční parametry volíme $\widetilde{\boldsymbol{\mu}}_0 = (0, \ldots, 0), \widetilde{\boldsymbol{\sigma}}_0 = (100, \ldots, 100)$.

Veškeré úlohy byly řešeny 10krát. Kompletní výsledky testů lze nalézt v příloze, viz sekce F. Zde uvedeme pouze průměrné časy a přesnosti nalezených řešení, viz tabulky 7.1 a 7.2.

Z výsledků testování lze vidět, že CE i FACE metoda nalezly vždy správné řešení. Také lze vidět, že v případě úlohy s velkým počtem lokálních extrémů, jakým je *trigonometrická* funkce, je řešení pomocí CE metody efektivnější než pomocí Matlab optimization toolboxu.

	CE		FACE		Matlab opt. t.	
n	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$
2	0.10	$2.33{\cdot}10^{-6}$	0.67	$2.23{\cdot}10^{-6}$	1.57	$1.03{\cdot}10^{-5}$
3	0.24	$8.83{\cdot}10^{-6}$	1.85	$1.17{\cdot}10^{-5}$	1.73	$1.25{\cdot}10^{-5}$
4	0.47	$3.66{\cdot}10^{-5}$	1.88	$6.04{\cdot}10^{-4}$	2.08	$1.39{\cdot}10^{-5}$
5	0.67	$7.58{\cdot}10^{-5}$	2.05	$8.43{\cdot}10^{-5}$	2.12	$1.46{\cdot}10^{-5}$
6	1.14	$7.68{\cdot}10^{-5}$	2.60	$9.27{\cdot}10^{-5}$	2.28	$1.58{\cdot}10^{-5}$
7	1.56	$1.03{\cdot}10^{-4}$	3.25	$1.48{\cdot}10^{-4}$	2.42	$1.60{\cdot}10^{-5}$
8	2.07	$2.76{\cdot}10^{-4}$	3.72	$7.13{\cdot}10^{-4}$	2.60	$1.54{\cdot}10^{-5}$
9	2.90	$2.82{\cdot}10^{-4}$	3.90	$4.98{\cdot}10^{-4}$	3.13	$1.62{\cdot}10^{-5}$
10	3.97	$3.39{\cdot}10^{-4}$	3.58	$2.68{\cdot}10^{-3}$	3.50	$1.62{\cdot}10^{-5}$

Tabulka 7.1: Výsledky testování pro *Rosenbrockovu* funkci

	CE		FACE		Matlab opt. t.	
n	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\overline{čas}$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$
2	0.10	$8.49{\cdot}10^{-7}$	0.61	$5.19{\cdot}10^{-7}$	1.03	0.0947
3	0.15	$4.18{\cdot}10^{-6}$	1.14	$3.12{\cdot}10^{-6}$	0.91	0.1894
4	0.26	$9.64{\cdot}10^{-6}$	1.82	$9.91{\cdot}10^{-6}$	1.36	0.5406
5	0.43	$1.82{\cdot}10^{-5}$	2.39	$2.20{\cdot}10^{-5}$	1.72	0.2837
6	0.69	$2.61{\cdot}10^{-5}$	2.99	$2.94{\cdot}10^{-5}$	1.95	0.4667
7	1.18	$3.76{\cdot}10^{-5}$	3.55	$4.51{\cdot}10^{-5}$	3.22	0.5556
8	1.49	$4.33{\cdot}10^{-5}$	4.12	$5.41{\cdot}10^{-5}$	3.87	0.6101
9	2.16	$5.54{\cdot}10^{-5}$	4.77	$6.78{\cdot}10^{-5}$	5.19	1.1835
10	3.09	$6.56{\cdot}10^{-5}$	5.32	$8.45{\cdot}10^{-5}$	3.69	1.4194

Tabulka 7.2: Výsledky testování pro *trigonometrickou* funkci

7.2 Úprava CE metody pro hladké optimalizace

V případě benchmarkových úloh z minulé sekce CE metoda nevyžívala hladkosti funkcí (možnost aproximace derivace). Nyní otestujeme kombinaci CE metody a metody pro hledání lokálních extrémů.

Tato kombinace bude využívat standardní CE metodu, viz algoritmus 5.5. Modifikací bude, že mezi 3. a 4. krokem metody (tedy po vygenerování a seřazení dle výkonnosti) použijeme nejlepších k snímků jako počáteční body metody pro hledání lokálních extrémů a nahradíme je výsledkem této metody.

Jako metodu pro hledání lokálních extrémů pro testování volíme funkci Matlabu `fminunc`, používající Quasi-Newtonovu metodu. Získali jsme tedy další dva volitelné parametry metody, počet snímků pro hledání lokálního extrému k a maximální počet iterací funkce `fminunc`. Tyto parametry mohou velmi ovlivnit rychlost konvergence me-

tody. Numerické testy ukázaly, že pro *Rosenbrockovu* funkci je vhodné klást větší důraz na metody pro hledání lokálních extrémů, tedy vyšší k a počet iterací, naopak při *trigonometrické* funkci stačí volit $k = 1$ a menší počet iterací, neboť slouží pouze pro zpřesnění již nalezeného optima.

V případě *Rosenbrockovy* funkce volíme parametry CE metody: $N = 1000, N_b = 20$, $\alpha_\mu = 0.8, \alpha_\sigma = 0.4$, $k = 2$ a maximální počet iterací 20. Výsledky testování pro $n = 10, 15$ a 20 lze vidět v následující tabulce. Všechny velikosti úlohy byly testovány 10krát, zaznamenaná hodnota $\overline{\text{poč.vyh. } H(x)}$ označuje průměrný celkový počet vyhodnocení funkce H (CE pokusy + `fminunc`).

n	$\overline{čas}$	$\overline{\text{poč.vyh. } H(x)}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\lVert \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rVert}$	$\max \lVert \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rVert$
10	2.32	56800	56.8	$1.68{\cdot}10^{-10}$	$7.30{\cdot}10^{-10}$	$2.06{\cdot}10^{-5}$	$5.14{\cdot}10^{-5}$
15	3.29	56600	56.6	$2.24{\cdot}10^{-10}$	$7.71{\cdot}10^{-10}$	$2.35{\cdot}10^{-5}$	$4.87{\cdot}10^{-5}$
20	4.42	56400	56.4	$8.65{\cdot}10^{-10}$	$1.88{\cdot}10^{-9}$	$5.26{\cdot}10^{-5}$	$8.67{\cdot}10^{-5}$

Tabulka 7.3: Výsledky pro minimalizaci *Rosenbrockovy* funkce

V případě *trigonometrické* funkce volíme parametry CE metody: $N = 1000, N_b = 20, \alpha_\mu = 0.8, \alpha_\sigma = 0.5$, $k = 1$ a maximální počet iterací 5. Výsledky testování pro $n = 10, 15$ a 20 lze vidět v následující tabulce. Všechny velikosti úlohy byly opět testovány 10krát.

n	$\overline{čas}$	$\overline{\text{poč.vyh. } H(x)}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\lVert \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rVert}$	$\max \lVert \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rVert$
10	0.89	62400	62.4	1.0000	1.0000	$2.62{\cdot}10^{-7}$	$9.60{\cdot}10^{-7}$
15	1.45	74600	74.6	1.0000	1.0000	$3.47{\cdot}10^{-7}$	$1.48{\cdot}10^{-6}$
20	2.07	83000	83	1.0000	1.0000	$3.45{\cdot}10^{-7}$	$1.35{\cdot}10^{-6}$

Tabulka 7.4: Výsledky pro minimalizaci *trigonometrické* funkce

Z výsledků testování lze vidět, že při hladkých optimalizacích je vhodné kombinovat CE metodu s metodami pro hledání lokálních extrémů.

8 Urychlení simulací použitím GPU paralelizace

Využití grafických karet (GPU) při Monte Carlo simulacích má veliký potenciál, neboť se téměř vždy jedná o *„Embarrassingly parallel"* problém (jeden MC pokus může vykonávat jedno vlákno), a tedy paralelní implementace je poměrně jednoduchá.

Jako technologie pro GPU implementaci vybraných úloh probíraných v této práci byla zvolena nVIDIA CUDA (základy lze najít v [20]), neboť umožňuje poměrně jednoduše (program je psán v jazyce C/C++) efektivní využití GPU. Jako prostředí, které zajišťuje programovou část na CPU byl zvolen Matlab. Tato kombinace byla optimální, neboť umožňuje zachování velké části předcházejícího sériového kódu (načítání dat úlohy a příprava dat), přičemž výpočetní část je rychle provedena na GPU.

8.1 Vlastnosti GPU

Nejprve krátce uveďme důležité vlastnosti GPU, které mají zásadní vliv na implementaci.

Oproti CPU, které má obvykle 4-16 výpočetních jader, má GPU vypočetních jader mnhohem více, obvykle v řádech tisíců. Výpočetní jádra na GPU jsou uspořádány do streamovacích multiprocesorů. V rámci jednoho bloku je možné mezi jádry komunikovat, jednotlivé bloky jsou oddělené. Jádra na GPU jsou však oproti jádrům na CPU slabší, a mají několik omezení

- malá cache[6] jádra (CPU může mít až několik MB, GPU má v řádech kB), toto způsobuje, že jádra GPU mohou efektivně pracovat s menším množstvím paměti najednou
- menší rychlost přístupu a stahování dat z paměti (bandwith), toto má za následek, že v případě úloh s velkou paměťovou náročností bude GPU málo efektivní
- pokud se nejedná o profesionální výpočetní GPU, pak může být počet jader, který umí počítat v dvojité přesnosti (double precision) malý. Proto při testování budeme používat pro GPU verzi pouze jednoduchou přesnost (single precision).

Veškeré testování bude provedeno na následujícím hardware.

Zařízení	Teoretický max. výkon	Počet jader	Max. bandwidth
nVIDIA GeForce GTX 760	2210 GFLOPS (single)	1152	192.2 GB/s
Intel Core i5-3570K	113.3 GFLOPS (double)	4	25.6 GB/s

Tabulka 8.1: Parametry testovacího hardware

Což znamená, že v případě stejně efektivní implementace na GPU získáme 19,5 krát rychlejší program.

[6] cache je druh paměti, který je přímo v jádře, v porovnání s klasickou pamětí RAM je mnohonásobně rychlejší

Dále uveďme základní tři typy paměti, které budeme při jednotlivých implementacích využívat, a jejich specifika

1. globální paměť - je největší dostupnou pamětí, čtení/zápis je možné ze všech vláken a je relativně pomalá
2. lokální paměť - je pouze pro jedno vlákno, má velmi malou velikost, základní rychlost je stejná jako u globální paměti, při správné práci s proměnnými a poli je GPU překopíruje do registrů (velice rychlá paměť)
3. paměť pro konstanty - oproti globální paměti má menší velikost, avšak je optimalizována pro skupinové čtení (více vláken čte stejnou pozici), zápis není při běhu programu povolen

Sdílenou paměť (shared memory) nebudeme potřebovat, neboť budeme využívat pouze přístup, kde jsou jednotlivá vlákna nezávislá.

Při spouštění programu se specifikuje, kolik bloků a kolik vláken v jednotlivých blocích poběží, přičemž vláken i bloků může být více než fyzických jader a streamovacích multiprocesorů.

Pro efektivní implementaci na GPU budeme muset dbát zejména na efektivní práci s pamětí, ostatní aspekty efektivity GPU jsou vyřešeny nezávislostí jednotlivých vláken. Důležitým faktem je, že čtení z paměti probíhá po skupinách prvků (*stride*), proto je nutné k paměti přistupovat po po sobě jdoucích prvcích.

8.2 GPU akcelerace CREC modelu

V případě tohoto modelu musela být vyřešena vysoká paměťová náročnost, která je způsobena dvěma faktory

- vstupní data úlohy jsou veliká (portfólio obsahuje ~3000 položek s 49 záznamy, data pro korelaci závazků, matice přechodu,...)
- pro možnost využití CE metody je třeba zachování vygenerovaných náhodných čísel (~7600 čísel na jeden pokus).

Dále bylo třeba implementovat inverzní distribuční funkci beta rozdělení, která není v matematickém balíčku pro CUDA dostupná.[7]

Jelikož použití CE metody klade na implementaci dodatečné nároky, byly vytvořeny dvě verze GPU implementace.

8.2.1 Implementace

Obě GPU implementace jsou téměř totožné, v případě implementace umožňující použití CE metody je však nutné CUDA kernel rozdělit na více částí a ukládat vygenerovaná náhodná čísla. Tato úprava způsobuje značné zpomalení simulace. Dále budeme popisovat implementaci s využitím CE metody.

[7] Implemetace inverzní beta distribuce byla inspirována C zdrojovým kódem dostupným na: http://people.sc.fsu.edu/~jburkardt/c_src/asa109/asa109.html

Jelikož jsou požadavky na paměť veliké, je nutné provádět simulaci po částech (dávky pokusů, které je možno s ohledem na velikost paměti provést najednou). Pro implementaci simulace bylo zvoleno použití více CUDA kernelů:

1. Kernel pro generování náhodných čísel, který generuje čísla z normálního rozdělení s danou střední hodnotou. Vygeneruje všechna potřebná náhodná čísla k provedení dávky pokusů.
2. Kernel pro spočtení LR, který spočte jednotlivá LR pokusů.
3. Kernel pro simulaci, který vyhodnotí ztrátu na základě vygenerovaných náhodných čísel.

Tyto kernely jsou následně volány v cyklu, dokud není proveden dostatečný počet pokusů.

Práce s pamětí: Jednotlivá vstupní data úlohy musela být přeuspořádána s ohledem na sekvenční přístup k paměti na GPU. Například náhodná čísla jsou generována po sloupcích, neboť je vyžadována jedna posloupnost náhodných čísel pro jeden řetězec, přičemž přistupovat k nim je třeba po řádcích. Toto je řešeno oddělením kernelu na generaci náhodných čísel a kernelu pro simulaci, kde je mezi vykonáním kernelů provedena transpozice.

Data, která mají konstantní velikost (v případě různých vstupních dat), jsou umístěna v konstantní paměti, jedná se o

- odmocnina matice korelací mezi sektory (část simulace korelace mezi závazky),
- matice přechodu (uvnitř simulace reprezentovány jako matice Z-score),
- matice odhadu krachu v budoucnu pro aproximaci zbytku řetězce a
- vektor koeficientů s odhadem inflace v budoucnu.

Všechna ostatní vstupní a výstupní data jsou umístěna v globální paměti. V globální paměti jsou také umístěna některá pole pro uložení dočasných hodnot, které jsou příliš velké pro lokální paměť.

8.2.2 Srovnání s implementací

Nyní si uveďme výsledky běhu programu pro 10^7 pokusů. Srovnáváme celkem 4 verze:

1. paralelní implementace v Matlabu na CPU (4 jádra),
2. paralelní implementace na GPU bez možnosti CE,
3. paralelní implementace na GPU s možností CE, avšak s původním rozdělením (žádný IS),

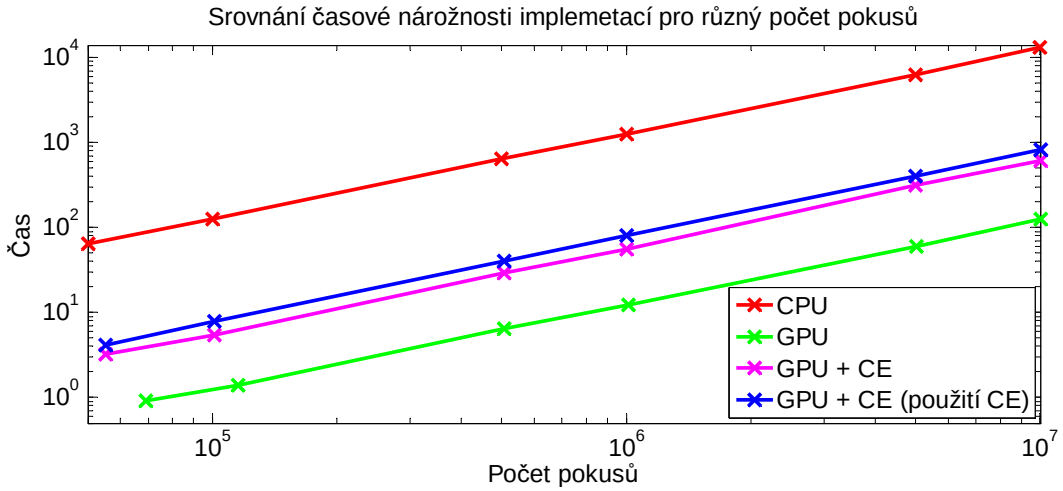
4. paralelní implementace na GPU s možností CE a jejím použitím (jediná verze používající IS).

CPU	GPU bez CE	GPU s CE (původní rozdělení)	GPU s CE (CE rozdělení)
12632 s	126s	647 s	877 s

Tabulka 8.2: Srovnání času běhů programu na CPU a GPU pro 10^7 pokusů

Jak lze vidět z výsledků testování, použití GPU výrazně urychlí výpočet. V případě implementace se stejnou funkčností jako CPU verze získáme 100 krát rychlejší výpočet, což značí i efektivnější využití dostupného hardware. V případě použití implementace umožňující CE metodu (a IS) je zrychlení pouze 19.5 krát, toto je způsobeno velkou paměťovou náročností implementace. Při užití IS spočteným pomocí CE metody získáme čas výpočtu 14,4 krát menší než CPU implementace, což je způsobeno různou délkou výpočtů jednotlivých řetězců (IS generuje více pokusů v oblasti velkých ztrát, jejichž výpočet trvá déle). Je nutné podotknout, že tento výsledek je mnohem přesnější (přibližně $330\times$ nižší rozptyl).

Dále uveďme graf škálovatelnosti jednotlivých verzí.



Obrázek 8.1: Škálovatelnost jednotlivých implementací

Jelikož se jedná o MC simulaci čas simulace roste lineárně s počtem pokusů.

Na závěr srovnejme celkovou efektivnost kombinace GPU + CE a to srovnáním jejich efektivnosti pomocí času na dosažení stejné přesnosti (zvolenou přesností je $RSO = 1\%$).

CPU	GPU bez CE	GPU s CE (původní rozdělení)	GPU s CE (CE rozdělení)
13651s	136s	699s	2.97 s

Tabulka 8.3: Srovnání odhadů času běhu implementací pro přesnost $RSO = 1\%$

Ačkoliv je GPU implementace umožňující použití CE metody výrazně pomalejší než implementace bez CE, konečná efektivita je velmi vysoká vzhledem k redukci rozptylu pomocí IS. Kombinací GPU a CE metody lze dosáhnout stejně přesného výsledku přibližně 4600krát rychleji než při užití paralelní CPU implementace.

8.3 GPU akcelerace úloh na kombinatorické optimalizace

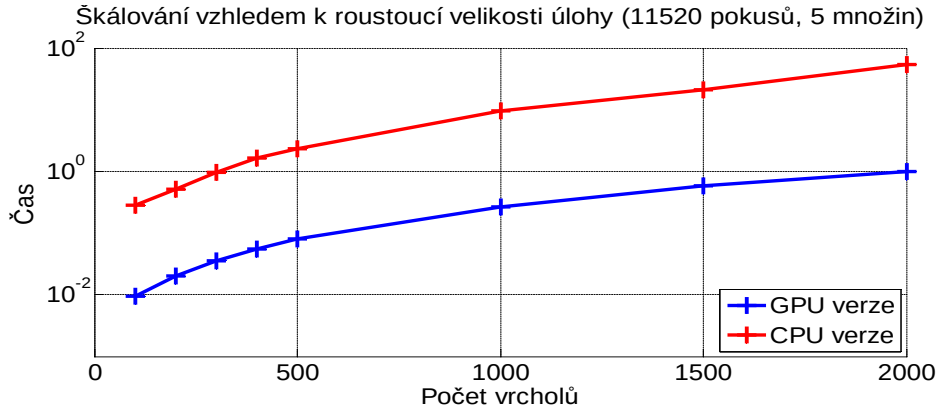
V této části se budeme zabývat GPU paralelizací optimalizačních úloh. Ve všech následujících úlohách budeme paralelizovat pouze generování náhodných pokusů a vyhodnocování výkonnostní funkce, neboť toto jsou časově nejnáročnější části CE algoritmu. Ve všech následujících srovnáních budeme pod pojmem CPU implementace rozumět sériovou implementaci používající pouze jedno vlákno. Teoretický nárůst výkonu pro výpočetně náročné úlohy je tedy 78krát a pro paměťově náročné úlohy 30krát.

8.3.1 Max-cut problém

V případě max-cut problému se jedná o velice paměťově náročnou úlohu, neboť při vyhodnocování výkonnostní funkce sčítáme značné množství hodnot váhové matice. Jelikož používáme přístup jednotlivých pokusů na jednotlivých vláknech, bude omezujícím faktorem rychlost čtení dat z matice vah, proto tuto matici umístíme do paměti pro konstanty. Tato paměť je však malá, najednou pojme maximálně matici 100×100 hodnot jednoduché přesnosti. Toto řešíme postupným načítáním 100×100 bloků váhové matice do konstantní matice a postupným sestavováním hodnoty výkonnostní matice z dostupných dat. Testování provedeme na syntetické úloze.

Porovnání škálování vzhledem k rostoucí dimenzi problému

Testování CPU a GPU implementace pro různé velikosti problému (zvyšující se počet vrcholů, dělení na 5 množin) lze vidět níže. Testujeme vždy čas nutný k generaci a vyhodnocení výkonnostní funkce pro 11520 pokusů (abychom mohli přesně srovnat obě implementace, neboť CUDA implementace umožňuje spouštění pokusů po dávkách o 2304 pokusech).



Obrázek 8.2: Porovnání škálování CPU a GPU implementace vzhledem k velikosti úlohy

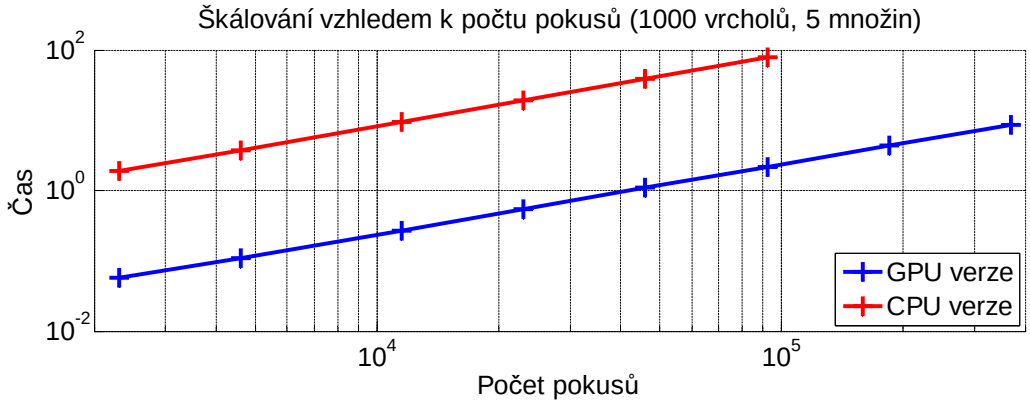
$\|V\|$	100	200	300	400	500	1000	1500	2000
CPU	0.2842	0.5167	0.9653	1.6331	2.3456	9.5835	20.8183	54.4837
GPU	0.0095	0.0199	0.0358	0.0557	0.0812	0.2699	0.5827	1.0025
poměr	29.79	25.95	26.97	29.29	28.88	35.51	35.72	54.35

Tabulka 8.4: Porovnání škálování CPU a GPU implementace vzhledem k velikosti úlohy

Z výsledků testování lze vidět, že GPU implementace zvyšuje svoji efektivnost s rostoucí dimenzí problému. Poměr zvýšení efektivity GPU implementace se pohybuje okolo teoretického zvýšení paměťové propustnosti.

Porovnání škálování vzhledem k zvyšujícímu se počtu pokusů

Dále testujeme čas nutný k vygenerování různého počtu pokusů a vyhodnocení jejich výkonnostní funkce pro 1000 vrcholů a dělení na 5 množin.



Obrázek 8.3: Porovnání škálování CPU a GPU implementace vzhledem k počtu pokusů

N	2304	4608	11520	23040	46080	92160	184320	368640
CPU	1.90	3.72	9.60	18.87	38.38	79.43	-	-
GPU	0.0568	0.1101	0.2699	0.5396	1.0843	2.1668	4.3165	8.6353
poměr	33.36	33.77	35.56	34.97	35.39	36.66	-	-

Tabulka 8.5: Porovnání škálování CPU a GPU implementace vzhledem k počtu pokusů

GPU efektivita s rostoucím objemem výpočetní práce lehce roste, neboť větší množství práce lze na GPU efektivněji rozdělit.

Srovnání výpočetního času pro celkové řešení úlohy

Pro úplné srovnání budeme řešit stejný problém pomocí CPU a GPU verze, výsledky srovnání pro 10 běhů programu lze nalézt v tabulce níže. Jedná se o syntetickou úlohu s 500 vrcholy a dělení na 5 množin.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	min H^*
CE-CPU	369	413	935000	1050000	74.8	84	100000	100000
CE-GPU	5.9	7.18	711936	870912	51.5	63	100000	100000

Tabulka 8.6: Srovnání CPU a GPU implementace pro řešení max-cut úlohy (10 běhů)

Mimo značně nižší čas výpočtu (62 krát nižší), si lze všimnout také menšího počtu iterací, který výrazně snížil dobu řešení úlohy. Toto je způsobeno vyšším počtem pokusů (12500/13824) v jednotlivých iteracích, což mělo za následek lepší konvergenci CE metody.

Test času a stability výpočtu pro rozsáhlejší úlohy

Na závěr testování ukážeme řešení několika rozsáhlých syntetických problémů pomocí GPU implementace. Výsledky pro první úlohu jsou průměry z 10 běhů, zbytek byl z časových důvodů testován pouze jednou.

počet vrcholů	počet množin	čas	celkový počet pokusů	iterace	H^*
1000	**5**	47.3	1802000	71.1	400000
2000	**10**	1576	17335296	171	1800000
4000	**2**	1104	3234816	78	4000000

Tabulka 8.7: Výsledky řešení GPU implementace CE metody pro různé velikosti úlohy

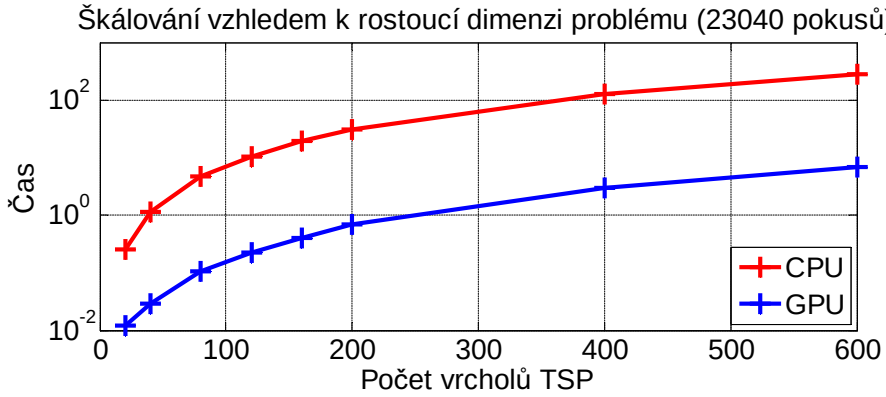
8.3.2 Problém obchodního cestujícího

V případě problému obchodního cestujícího se opět jedná o velice paměťově náročnou úlohu, neboť při generaci Markovova řetězce bez opakování potřebujeme v každém vlákně při každém pokusu přečíst celou matici přechodu. V tomto případě však není možnost ukládání matice přechodu do paměti pro konstanty, neboť potřebujeme všechna data najednou.

Vyhodnocení výkonnostní funkce je provedeno pomocí nativních funkcí Matlabu pro práci na GPU, neboť se jedná o jednoduchý problém, který lze řešit vektorovými operacemi. Testování provedeme na syntetické úloze.

Porovnání škálování vzhledem k rostoucí dimenzi problému

Testování CPU a GPU implementace pro různé počty vrcholů lze vidět níže. Opět testujeme čas nutný k generaci a vyhodnocení výkonnostní funkce, tentokrát pro 23040 pokusů .



Obrázek 8.4: Škálování CPU a GPU implementace vzhledem k velikosti úlohy

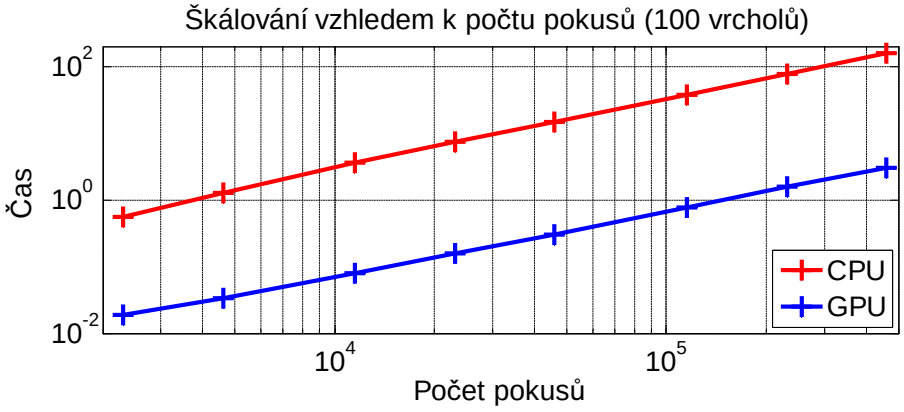
$\|V\|$	**20**	**40**	**80**	**120**	**160**	**200**	**400**	**600**
CPU	0.25	1.13	4.61	10.58	19.18	30.25	125.23	282.64
GPU	0.0119	0.0289	0.1052	0.2222	0.4077	0.6768	2.9280	6.8545
poměr	21.41	39.24	43.79	47.62	47.05	44.70	42.77	41.23

Tabulka 8.8: Škálování CPU a GPU implementace vzhledem k velikosti úlohy

Z výsledků testování opět vidíme, že GPU implementace zvyšuje svoji efektivnost s rostoucí dimenzí problému. V případě úlohy s 600 vrcholy je poměr efektivity GPU ku CPU téměř dvojnásobný oproti úloze s 20 vrcholy.

Porovnání škálování vzhledem k zvyšujícímu se počtu pokusů

Dále testujeme čas nutný k vygenerování různého počtu pokusů a vyhodnocení jejich výkonnostní funkce pro 100 vrcholů.



Obrázek 8.5: Škálování CPU a GPU implementace vzhledem k počtu pokusů ($n = 100$)

N	**2304**	**4608**	**11520**	**23040**	**46080**	**115200**	**230400**	**460800**
CPU	0.55	1.26	3.58	7.32	14.61	37.02	78.16	159.40
GPU	0.0185	0.0331	0.0790	0.1545	0.3046	0.7627	1.5644	3.0643
poměr	29.50	38.17	45.27	47.38	47.96	48.53	49.97	52.02

Tabulka 8.9: Škálování CPU a GPU implementace vzhledem k počtu pokusů ($n = 100$)

Opět lze zaznamenat rostoucí efektivitu GPU implementace při zvyšování počtu pokusů. Tentokrát je však zlepšení mnohem znatelnější.

Srovnání výpočetního času pro celkové řešení úlohy

Pro úplné srovnání budeme řešit stejný problém pomocí CPU a GPU verze, výsledky srovnání pro 10 běhů programu lze nalézt v tabulce níže. Jedná se o syntetickou úlohu s 80 vrcholy.

metoda	$\overline{čas}$	$\max čas$	$\overline{N_{sum}}$	$\max N_{sum}$	$\overline{Iter}$	$\max Iter$	$\overline{H^*}$	$\max H^*$
CE-CPU	309	353	1593600	1856000	49.8	58	82.83	85.87
CE-GPU	7.33	8.52	1664400	2032128	51.6	63	82.92	85.43

Tabulka 8.10: Srovnání CPU a GPU impl. pro řešení syntetické TSP pro 80 měst (10 běhů)

Výsledku bylo dosaženo za průměrně 42 krát nižší čas.

Test času a stability výpočtu pro rozsáhlejší úlohy

Na závěr testování ukážeme řešení několika rozsáhlých syntetických problémů pomocí GPU implementace.

počet vrcholů	**čas**	**celkový počet pokusů**	**iterace**	H^*
100	23.97	3598848	71	103.29
150	286.07	16821504	149	156.76
200	798.6	24454656	122	214.3

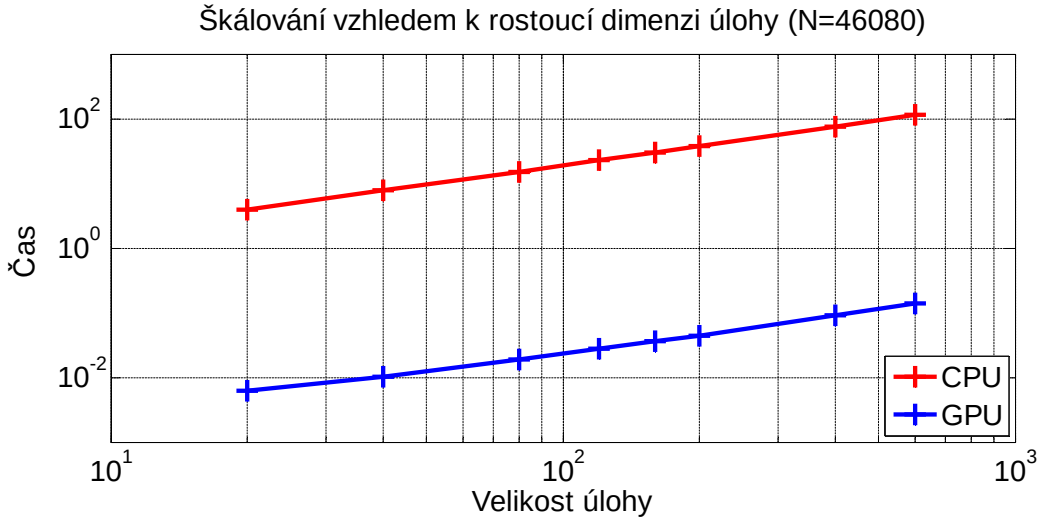
Tabulka 8.11: Výsledky řešení GPU implementace CE metody pro různé velikosti úlohy

8.3.3 Problém zarovnání sekvencí

V případě problému zarovnání sekvencí se opět setkáváme velkou matici přechodu, kterou nelze umístit do paměti pro konstanty. Toto však oproti GPU implementaci problému obchodního cestujícího nemá takový vliv na výkon, neboť v generaci jednoho snímku potřebujeme přečíst pouze zlomek této matice. Generování i vyhodnocování výkonnostní funkce je provedeno zároveň v jednom CUDA kernelu. Testování provedeme na syntetické úloze tvořené dvěma náhodnými sekvencemi o délkách $n = n_1 = n_2$ nad abecedou hodnot $\Sigma = \{1, 2, \ldots, 10\}$.

Porovnání škálování vzhledem k rostoucí dimenzi problému

Testování CPU a GPU implementace pro různé velikosti n lze vidět níže. Testujeme opět vždy čas nutný k generaci a vyhodnocení výkonnostní funkce pro 46080 pokusů.



Obrázek 8.6: Škálování CPU a GPU implementace vzhledem k velikosti úlohy

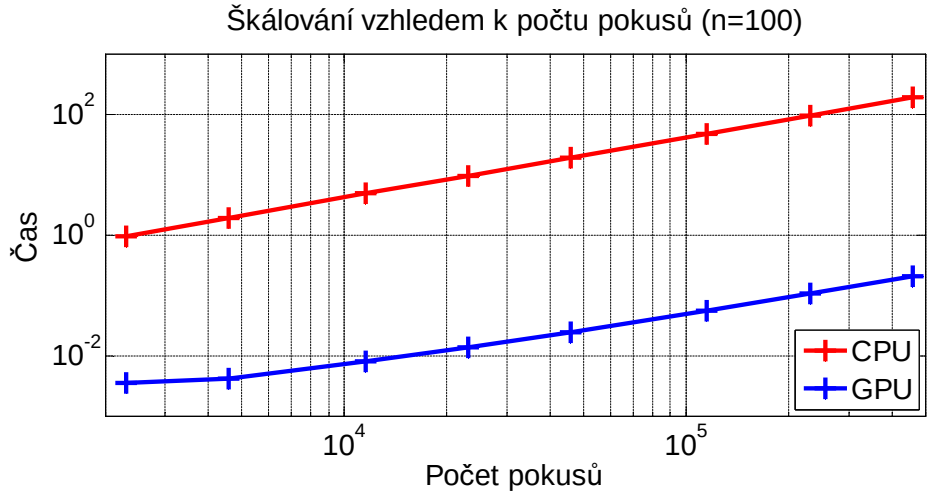
$\|V\|$	**20**	**40**	**80**	**120**	**160**	**200**	**400**	**600**
CPU	3,89	7,66	15,24	22,73	30,43	37,86	76,32	115,02
GPU	0,0062	0,0103	0,0187	0,0275	0,0364	0,0442	0,0900	0,1398
poměr	629,5	741,3	814,7	825,3	835,3	857,3	848,0	822,8

Tabulka 8.12: Škálování CPU a GPU implementace vzhledem k velikosti úlohy

Z výsledků sledujeme oproti předchozím implementacím značně lepší poměr efektivity GPU ku efektivitě CPU. Toto je způsobeno jednak pomalou CPU implementací, kde vzhledem k Markovovu řetězci o předem neznámé délce nebylo možné využít vektorizaci výpočtů, a lepším využitím GPU, neboť se nejedná o tak paměťově náročnou úlohu. Také stejně jako v předchozích implementacích se s rostoucí velikosti problému zlepšuje GPU ku CPU efektivity.

Porovnání škálování vzhledem k zvyšujícímu se počtu pokusů

Dále testujeme čas nutný k vygenerování různého počtu pokusů a vyhodnocení jejich výkonnostní funkce pro $n = 100$.



Obrázek 8.7: Škálování CPU a GPU implementace vzhledem k počtu pokusů ($n = 100$)

N	**2304**	**4608**	**11520**	**23040**	**46080**	**115200**	**230400**	**460800**
CPU	0,96	1,92	4,78	9,51	19,09	47,52	95,40	191,33
GPU	0,0036	0,0041	0,0081	0,0138	0,0238	0,0549	0,1054	0,2057
poměr	268,3	466,7	588,7	689,6	802,9	866,2	904,8	930,3

Tabulka 8.13: Škálování CPU a GPU implementace vzhledem k počtu pokusů ($n = 100$)

Opět lze zaznamenat, že poměr efektivity GPU a CPU implementace s rostoucí velikostí výpočetní úlohy značně roste.

Srovnání výpočetního času pro celkové řešení úlohy

Pro úplné srovnání CPU a GPU implementace budeme řešit úlohu z příkladu 6.8.

metoda	$\overline{čas}$	max $čas$	$\overline{N_{sum}}$	maxN_{sum}	$\overline{Iter}$	max $Iter$	$\overline{H^*}$	max H^*
CE-CPU	93.7	99.3	166840	175616	13.3	14	36	36
CE-GPU	0.15	0.17	166260	177408	13.12	14	36	36

Tabulka 8.14: Srovnání CPU a GPU implementace pro řešení úlohy 6.8 (10 běhů)

Při použití GPU implementace bylo výsledku dosaženo 624 krát rychleji.

9 Závěr

Hlavním cílem této práce bylo studium Cross-entropy metody a její aplikace na optimalizační úlohy z oblasti spojitých optimalizací a složitých kombinatorických problémů.

V teoretické části této práce jsme se nejprve seznámili s důležitou metodou pro redukci rozptylu *importance sampling*. Na jednoduché motivační úloze jsme pak postupně odvodili optimální hustotu pravděpodobnosti, *variance minimization* metodu a Cross-entropy metodu, ukázali jsme také srovnání těchto přístupů včetně jejich numerického řešení pomocí *stochastic counterpart* přístupu. Dále jsme ukázali, jak využít Cross-entopy metodu pro *rare events (RE)* úlohy a na závěr jsme ukázali převedení optimalizační úlohy na asociovaný stochastický problém a jeho řešení Cross-entropy metodou.

V průběhu tvorby této práce byla také možnost spolupráce s finanční institucí a využití probírané tématiky na úloze z praxe. Jedná se konkrétně o CREC model, který se využívá v bankovnictví pro stanovení finanční rezervy nutné pro vyrovnání případných ztrát a dá se zařadit mezi RE systémy. Simulace tohoto modelu byla velmi časově náročná, proto byl tento model nejprve paralelizován pomocí GPU a následně bylo použito *importance sampling* pomocí Cross-entropy metody. Pro úspěšné použití Cross-entropy musela být metoda modifikována, neboť se jedná o rozsáhlý model s velkým počtem parametrů. Byla použita adaptivní verze Cross-entopy metody zkombinovaná s adaptivním vyhlazovacím parametrem a *screening* metodou. IS nalezený touto metodou dosáhl 321krát menšího rozptylu. V kombinaci s GPU paralelizací lze oproti původní implementaci dosáhnout výsledku přibližně 4600krát rychleji.

Dále v praktické části této práce byly řešeny pomocí Cross-entropy metody tři problémy na kombinatorickou optimalizaci, konkrétně problém obchodního cestujícího, max-cut problém a problém zarovnávání sekvencí. U všech těchto problémů jsme řešili velikosti úloh, které by nebyly analytickou cestou řešitelné. Pro možnost řešení rozsáhlých úloh byla také provedena paralelní implementace na GPU. Pro srovnání s jinými simulačními technikami pro řešení kombinatorických optimalizací bylo provedeno testování metod Cross-entropy, genetických algoritmů, simulovaného žíhání a mravenčích kolonií na problému obchodního cestujícího. Přičemž Cross-entropy metoda dosahovala srovnatelných výsledků s ostatními metodami, navíc byla provedena modifikace Cross-entropy metody zahrnující charakter problému, která dosáhla mezi porovnávanými nejlepších výsledků.

Také byla vyzkoušena CE metoda v aplikaci na spojité optimalizace, srovnána se softwarem Matlab pro globální optimalizace. V případě problémů s mnoha lokálními extrémy Cross-entropy metoda oproti Matlabu vždy nalezla optimální řešení. Dále byla navržena kombinace CE metody s metodami pro hledání lokálních extrémů při řešení hladkých optimalizací, konkrétně Quasi-Newtonovou metodou. Tato kombinace byla otestována na stejných spojitých optimalizačních problémech, přičemž dosáhla značně lepších výsledků.

Možným pokračování práce je využití Cross-entopy metody pro hledání podobností v rozsáhlých souborech medicínských dat (geny v DNA řetězcích), za účelem identifikace faktorů způsobujících rakovinu.

10 Literatura

[1] RUBINSTEIN, Reuven Y.; KROESE, Dirk P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004. ISBN 978-0-387-21240-1.

[2] MEERSCHAERT, Mark M. *Mathematical modeling*. Academic press, 2013. ISBN 978-0-12-386912-8.

[3] RUBINSTEIN, R. Y.; SHAPIRO A. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. John Wiley & Sons, Inc., 1993. ISBN 978-0-471-93419-6.

[4] KROESE, Dirk P.; TAIMRE, Thomas; BOTEV, Zdravko I. *Handbook of Monte Carlo Methods*. John Wiley & Sons, Inc., 2011. ISBN 978-0-470-17793-8

[5] RUBINSTEIN, Reuven Y.; KROESE, Dirk P. *Simulation and the Monte Carlo method*. John Wiley & Sons, Inc., 2008. ISBN 978-0-470-17794-5.

[6] COSTA, Andre; JONES, Owen Dafydd; KROESE, Dirk. *Convergence properties of the cross-entropy method for discrete optimization*. *Journal Operations Research Letters*. Elsevier Science Publishers B. V., 2007, (Volume: 35, Issue: 5) str. 573-580. ISSN 0167-6377.

[7] KROESE, Dirk P.; RUBINSTEIN, Reuven Y.; GLYNN, Peter W. *The cross-entropy method for estimation. Handbook of Statistics, Volume 31: Machine Learning Theory and Applications*, North Holland & IFIP, 2013. ISBN 978-0-444-53859-8

[8] DAVIS, Lawrence, et al. (ed.). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991. ISBN 978-0-442-00173-5.

[9] AARTS, Emile; KORST, Jan. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc., 1989. ISBN 0-471-92146-7.

[10] DORIGO, Marco; MANIEZZO, Vittorio; COLORNI, Alberto. *Ant system: optimization by a colony of cooperating agents*. *Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE Transactions on, 1996, (Volume: 26, Issue: 1) str. 29-41. ISSN 1083-4419.

[11] KEITH, Jonathan; KROESE, Dirk P. *Rare event simulation and combinatorial optimization using cross entropy: sequence alignment by rare event simulation*. Winter Simulation Conference, 2002. str. 320-327. ISBN 0-7803-7615-3

[12] LITSCHMANNOVÁ, Martina. *Úvod do statistiky*, [online]. Ostrava, 2011 [cit. 2015 -04-03]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/uvod_do_statistiky.pdf

[13] BRIŠ, Radim; LITSCHMANNOVÁ, Martina. *Statistika II*. Ostrava: Vysoká škola báňská - Technická univerzita, 2007, 1 CD-R. ISBN 978-80-248-1482-7.

[14] LITSCHMANNOVÁ, Martina. *Vybrané kapitoly z pravděpodobnosti*, [online]. Ostrava, 2011 [cit. 2015-01-03]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/vybrane_kapitoly_pravdepodobnost.pdf

[15] BOUCHALA, Jiří. *Variační metody*, [online]. Ostrava, 2012 [cit. 2015-04-03]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/variacni_metody.pdf

[16] KOVÁŘ, Petr. *Teorie grafů*, [online]. Ostrava, 2012 [cit. 2015-02-02]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/skriptum_teorie_grafu_rozsirene_tisk.pdf

[17] DUBI, Arie. *Monte Carlo applications in systems engineering*. John Wiley & Sons, Inc., 1999. ISBN 0-471-98172-9.

[18] YUSOF, N.M.; JAFFAR, Maheran Mohd. *Predicting the credit risk through Merton model*. *Business, Engineering and Industrial Applications (ISBEIA)*, IEEE Transactions on, 2011, str. 162 - 166. ISBN 978-1-4577-1548-8

[19] CHEUNG, K. Y.; LEE, Stephen MS. *Variance estimation for sample quantiles using the m out of n bootstrap*. Annals of the Institute of Statistical Mathematics, 2005, (Volume: 57, No.: 2), str. 279-290.

[20] KIRK, David, et al. *NVIDIA CUDA software and GPU parallel computing architecture*, [online] In: ISMM. 2007 [cit. 2015-03-04]. Dostupné z: http://kr.nvidia.com/content/cudazone/download/showcase/kr/Tutorial-DKIRK.pdf

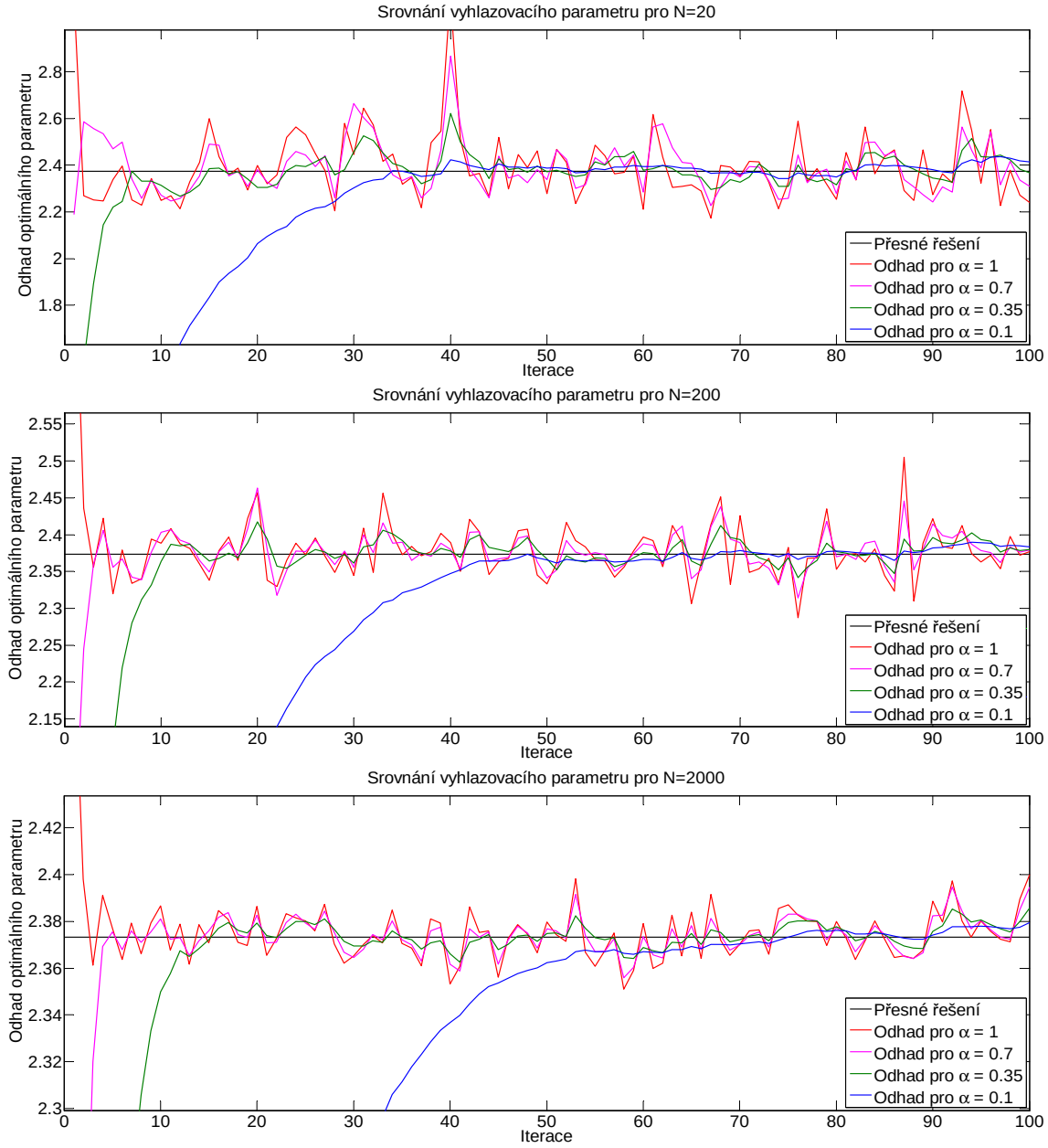
[21] PÁTEK, Zdeněk. *Multiple sequence alignment pomocí genetických algoritmů*. Diplomová práce, Universita Karlova, 2012.

[22] UNIPROT CONSORTIUM, et al. *The universal protein resource (UniProt)*. Nucleic acids research, 2015. Dostupné z: http://www.uniprot.org/

[23] DOSTÁL, ZDENĚK; BERMELIJSKI, P. *Metody optimalizace*, [online]. Ostrava, 2012 [cit. 2015-04-01]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/metody_optimalizace.pdf

A Numerické testování vyhlazovacího parametru pro CE metodu

Pro jednoduchý numerický test vyhlazovacího parametru použijeme úlohu 3.1. Budeme testovat tři počty pokusů $N = 20, 200, 2000$ a čtyři velikosti vyhlazovacích parametrů $\alpha = 1, 0.7, 0.35, 0.1$. Výsledky lze vidět na obrázcích níže.

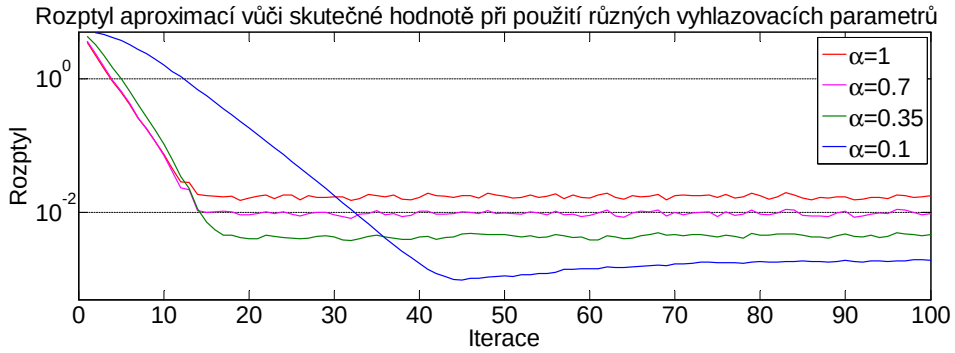


Obrázek A.1: Ukázka vlivu vyhlazovacího parametru na konvergenci algoritmu 3.29

Z výsledků testu vlivu vyhlazovacího parametru lze vyvodit několik pozorování:

- Při využití menšího α můžeme dosáhnout přesnější aproximace, algoritmus však bude potřebovat více iterací.
- Rozptyl aproximací se snižuje s lineárně vzhledem k N.

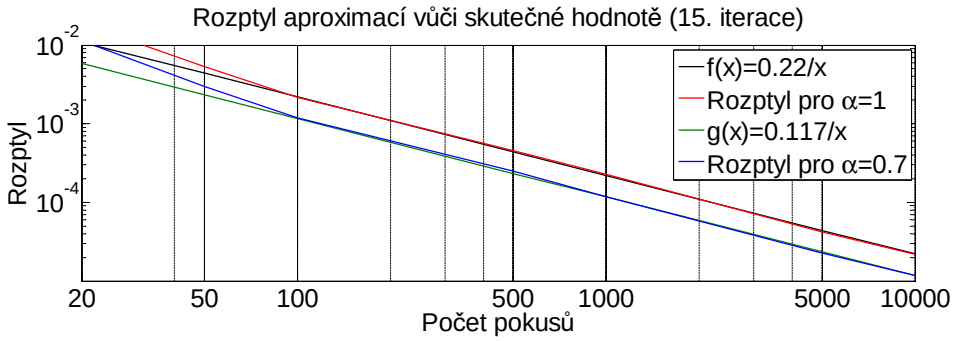
Tyto pozorování numericky ověříme na testovací úloze. Provedeme několik běhů metody (konkrétně 1000) a spočítáme rozptyl odhadu metody v jednotlivých krocích (výběrový rozptyl vůči skutečné hodnotě parametru). Nejprve pro ověření prvního pozorování spočtěme rozptyly pro konstantní $N = 20$ a různá α.



Obrázek A.2: Rozptyl aproximace v závislosti na iteraci a vyhlazovacím parametru

Jak lze vidět na obrázku výše, skutečně se snižujícím se α získáváme přesnější odhad bez nutnosti navyšování počtu pokusů. Také lze vidět, že k dosažení konvergence metody pro nízká α je třeba mnohem více iterací.

Pro ověření druhého pozorování provedeme podobný test avšak pro pevnou iteraci (15.) a měnící se počet pokusů.



Obrázek A.3: Rozptyl aproximace v závislosti na počtu pokusů

Jak lze vidět z výsledků testu, pro vyšší počty pokusů se skutečně rozptyl snižuje lineárně. Je také vidět, že redukce rozptylu způsobená volbou α je nezávislá na počtu pokusů

B Popis CREC modelu

Jak už bylo zmíněno, CREC model slouží ke stanovení finanční rezervy pro pokrytí případných ztrát způsobených insolvencí protistran.

Model je založen na generování ekonomických scénářů a vyhodnocením výsledné potencionální ztráty. Tyto ztráty pak tvoří distribuční funkci, jejíž 99.995% kvantil nás zajímá. Tento kvantil se v praxi nedá spočítat analyticky a proto je určen přibližně pomocí Monte Carlo metody.

Pro pochopení modelu je třeba uvést několik důležitých pojmů.

Závazek *(„Exposure")* - Je základem modelu, jedná se o příslušnou velikost maximální ztráty v případě krachu (*„Exposure at default"* (EAD)) v daném čase. Jednotlivé závazky jsou mezi sebou silně korelované, tato korelace je dána dvěma faktory: příslušností k sektoru a vazbami způsobenými příslušností k jedné společnosti. Může mít také příslušnost k nějakému státu.

Sektor - Udává souhrnně příslušnost závazku nebo státu k odvětví a regionu (např. průmysl v Asii, Severní Amerika, ...).

Společnost *(„Legal entity")* - Zajišťuje vazby mezi závazky způsobené jednou společností nebo koncernem. Jsou ve stromové struktuře se danou mírou závislosti.

Známky *(„Rating")* - Značí spolehlivost daného závazku v závislosti na úvěruschopnosti protistrany. V modelu je použito hodnocení od 1 (bez rizika) do 18 (v krachu).

Model je složen ze základních dvou částí: generace ekonomického scénáře a jeho vyhodnocení.

B.1 Generování ekonomického scénáře

V ekonomickém scénáři nás zajímají známky závazků a státu po jednom roce. Ten je generován jako jeden krok Markovova řetězce, přičemž sledovanou hodnotou jsou právě jednotlivé známky závazků a států. Veškeré korelace v modelu jsou uvažovány v normální kopuli. Proto jsou přechody simulovány pomocí náhodných čísel z $N(0,1)$, které se v rámci modelu nazývají šoky.

Jako vstupní data do modelu přichází:

- korelační matice pro jednotlivé sektory;
- hierarchická strukturu společností s jednotlivými korelačními koeficienty;
- jednotlivé závazky s příslušností k sektoru (s příslušnou korelací), společnosti (s příslušnou korelací) a státu, počáteční známkou a maticí přechodu;
- státy s příslušností k sektoru (s příslušnou korelací), počáteční známkou a maticí přechodu.

Generování ekonomického scénáře se dá rozdělit do několika částí.

B.1.1 Generování šoků pro sektory

Máme zadáno $S = (s_1, \ldots, s_n)$ sektorů, jejichž šoky jsou z vícerozměrného normálního rozdělení daného korelační maticí C, tedy jednotlivé střední hodnoty $\mu_i = 0$ a rozptyly jednotlivých složek jsou $\sigma_i = 1$. Náhodný pokus z tohoto rozdělení spočteme jako:

$$\overline{X_S} = L \cdot Z,$$

kde L je odmocnina matice C $\left(C = LL^T\right)$ (spočtená například Choleského rozkladem) a $Z = (z_i, \ldots, z_n) \sim N(0, 1)$.

B.1.2 Generování šoků pro společnosti

Máme dáno $L = (l_1, \ldots, l_m)$ společností uspořádaných do hierarchické stromové struktury. Náhodný šok společnosti l_i spočteme jako

$$(X_L)_i = \sqrt{1 - \omega_i^2} \cdot Z + \omega_i \cdot (X_L)_j \,, \tag{B.1}$$

kde $Z \sim N(0, 1)$, $(X_L)_j$ je šok nadřazené společnosti (pokud taková existuje) a ω_i udává korelaci s nadřazenou společností l_j. V případě, že l_i nemá nadřazenou společnost, je

$$(X_L)_i = Z.$$

Ze vzorce B.1. je jasné, že se šoky musí počítat s ohledem na hierarchickou strukturu (od kořene k listům).

B.1.3 Generování šoků pro závazky

V případě závazků je třeba zahrnout korelaci se sektorem i se společností, což je v modelu vyřešeno ve dvou krocích:

$$\begin{aligned} (X_E)'_i &= \sqrt{1 - \varepsilon_i^2} \cdot Z + \varepsilon_i \cdot (X_S)_k \,, \\ (X_E)_i &= \sqrt{1 - \omega_i^2} \cdot (X_E)'_i + \omega_i \cdot (X_L)_j \,, \end{aligned}$$

kde $Z \sim N(0, 1)$, $(X_S)_k$ je šok sektoru, do nějž závazek patří, ε_i udává korelaci se sektorem, $(X_L)_j$ je šok společnosti, které závazek patří a ω_i udává korelaci se společností.

B.1.4 Generování šoků pro státy

Šoky pro státy se generují stejným způsobem jako předešlé. Šok státu c_i získáme jako

$$(X_C)_i = \sqrt{1 - \varepsilon_i^2} \cdot Z + \varepsilon_i \cdot (X_S)_k \,,$$

kde $Z \sim N(0, 1)$, $(X_S)_k$ je šok sektoru, do nějž stát patří, a ε_i udává korelaci se sektorem.

B.1.5 Spočtení výsledné známky

V tuto chvíli již máme spočtený šok $X \sim N(0,1)$, máme dánu počáteční známku g a matici přechodu P. Známka určuje příslušný řádek v matici, jenž nám dává přechodové pravděpodobnosti. Spočteme-li hodnotu distribuční funkce pro daný šok

$$p = F(X),$$

pak příslušnou známku získáme jako nejmenší k, pro které platí:

$$p < \sum_{i=1}^{k} (P)_{g,i}.$$

B.2 Vyhodnocení celkové ztráty

Celková ztráta je součtem ztrát způsobených jednotlivými závazky. Pro každý závazek může nastat jedna ze tří situací generující ztrátu:

- krach závazku, který nastane v případě, že nová známka závazku je 18;
- krach závazku v důsledku krachu státu, který nastane v případě, že závazek má známku menší než 18 a stát má známku 18;
- předpověď ztráty v období splatnosti závazku, která se počítá ve všech ostatních případech.

B.2.1 Ztráta v případě krachu závazku

V případě krachu závazku banka neztratí celou částku, ale pouze její část. Tato část se nazývá *„Loss given at default"* (LGD). V tomto modelu se LGD řídí beta rozdělením, jehož parametry jsou specifické pro každý závazek.

LGD je stejně jako náhodný šok korelováno se společností, jíž závazek patří, toto je simulováno stejně jako v případě šoků a výsledná realizace z beta rozdělení je dána jako

$$LGD_i = B^{-1}(F(X)),$$

kde X je korelovaný náhodný pokus, F je distribuční funkce $N(0,1)$ a B^{-1} je inverze distribuční funkce beta rozdělení daného specifickými parametry závazku.

Ztrátu v případě krachu pak získáme jako součin LGD_i a maximální výše ztráty.

B.2.2 Ztráta v případě krachu státu

V případě krachu státu postupujeme stejně, avšak LGD_i je z beta rozdělení daného specifickými parametry státu.

B.2.3 Odhad ztráty v období splatnosti závazku

V případě, že nenastal krach závazku ani státu, je odhadována ztráta, která může nastat v období druhého roku až do splatnosti závazku. Pro výpočet tohoto odhadu využijeme následující vstupní data:

- maximální výše ztráty závazku v daném roce;
- střední hodnoty rozdělení LGD závazku a státu do kterého závazek patří;
- pravděpodobnosti krachu státu nebo závazku v daném roce;
- změna hodnoty měny v závislosti na inflaci v budoucnosti.

Odhad ztráty pak spočteme jako součet odhadů ztrát v jednotlivých letech, které získáme jako

$$\begin{aligned} Z(t) &= EAD(t) \cdot I(t) \cdot \mu_E \cdot \phi_E(t) \cdot \left(1 - \sum_{i=2}^{t} \phi_C(t)\right) \\ &\quad + EAD(t) \cdot I(t) \cdot \mu_C \cdot \phi_C(t) \cdot \left(1 - \sum_{i=2}^{t} \phi_E(t)\right) \\ &\quad + EAD(t) \cdot I(t) \cdot \mu_E \cdot \phi_E(t) \cdot \phi_C(t), \end{aligned}$$

kde $\phi(t)$ je pravděpodobnost krachu závazku/státu v roce t, μ je střední hodnota LGD závazku/státu, $EAD(t)$ je maximální ztráta závazku v roce t a $I(t)$ je změna hodnoty závazku v důsledku inflace do roku t. Lze vidět, že tato ztráta je tvořena třemi částmi:

1. ztráta v důsledku krachu závazku (v případě, že stát nezkrachoval);
2. ztráta v důsledku krachu státu;
3. ztráta v případě, že zkrachuje závazek i stát.

C Výpisy zdrojových kódů

```
1  function [ vysledek, argument, num_iter, N_sum ] = CE(generate, H, update,
       v_0, N, N_b, alpha, d)
2  %%
3  % Funkce pro optimalizaci funkce H pomocí CE metody
4  %
5  % generate - X=generate(v,N), funkce pro generaci pokusů
6  % H - H_X=H(X), funkce pro spočtení výkonnosti pokusů
7  % update - v=update(v,alpha,X,H_sort,index,N_b), funkce pro aktualizaci
     parametrů
8  % v_0 - cektor počátečních parametrů
9  % N - počet pokusů v jedné iteraci CE metody
10 % N_b - počet elitních snímků požitých pro aktualizaci parametrů rozdělení
11 % alpha - tlumící parametr při aktualizaci parametrů rozdělení
12 % d - počet iterací se stejnou maximální hodnotou pro ukončení metody
13 %%
14
15 v=v_0;
16 H_old=-inf*ones(1,d);
17 vysledek=-inf;
18 N_sum=0;
19 for i=1:1000
20    X=generate(v,N); % generování náhodných pokusů
21    N_sum=N_sum+N;
22    H_X=H(X); % spočtení výkonností
23    [H_sort,index]=sort(H_X); %seřazení od nejmenšího po nejvyšší
24    gamma=H_sort(N-N_b+1);
25    H_max=H_sort(end); % elitní snímek
26    H_old=[H_old(2:end) H_max]; % buffer předchozích nej. hodnot
27    if H_max>vysledek
28       vysledek=H_max; %doposud nejlepší hodnota
29       argument=X(index(end),:); % argument doposud nej. výsledku
30    end
31    v=update(v,alpha,X,H_sort,index,N_b); % update parametrů dle N_b elitní
        ch snímků
32    if same_values([H_max H_old]) %kontrola stagnace v optimálním řešení
33       disp(['It:' num2str(i) ', celkem pokusů:' num2str(N_sum) '. Nalezeno
           maximum o hodnotě: ' num2str(vysledek)]);
34       num_iter=i;
35       break;
36    end
37 end
38 end
```

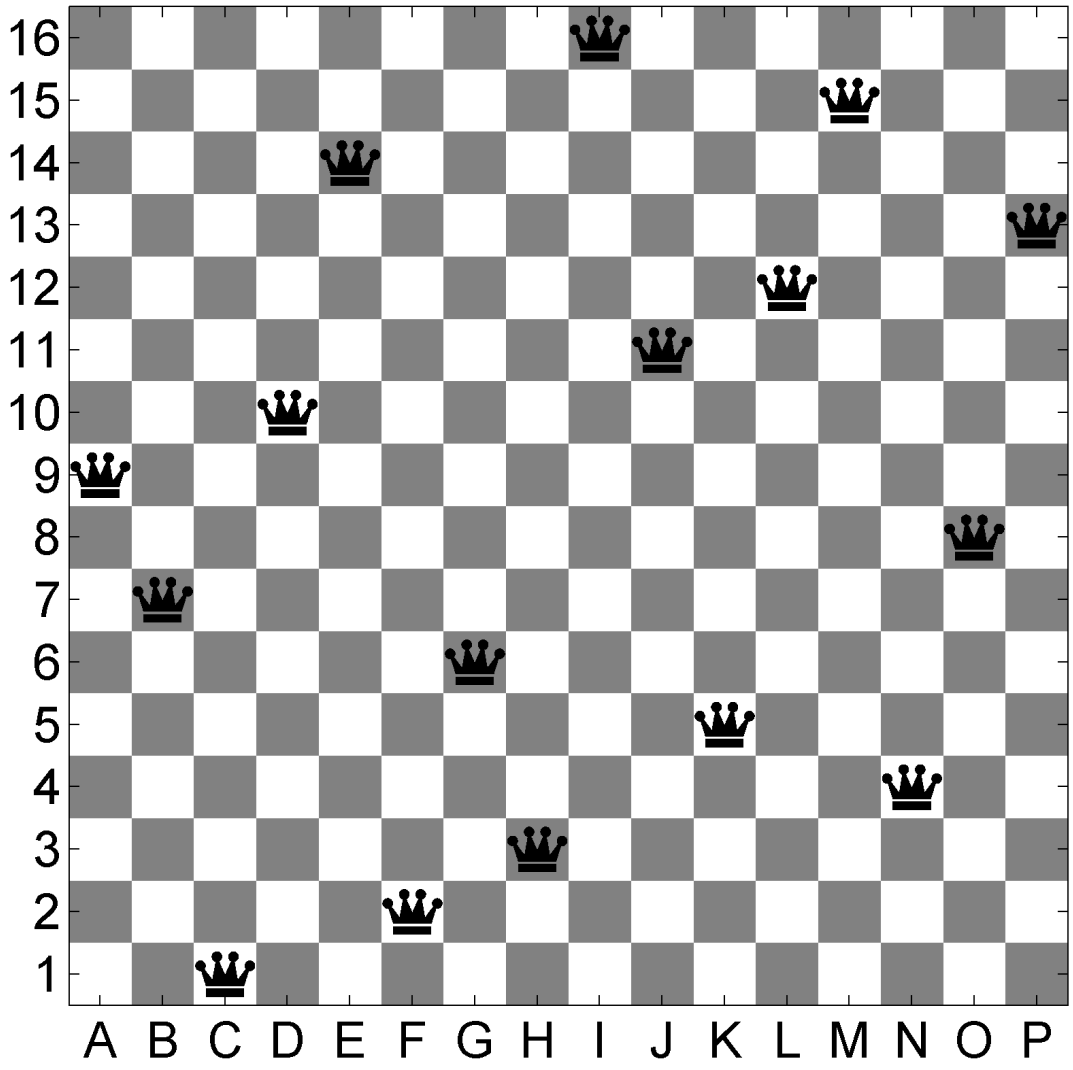
Výpis 1: CE metoda pro optimalizace

```
1  function [ vysledek, argument, num_iter, N_sum ] = FACE(generate, H,
     update, v_0, N_min, N_max, beta, N_b, alpha, c, d)
2  %%
3  % Funkce pro optimalizaci funkce H pomocí FACE metody
4  %
5  % generate - X=generate(v,N), funkce pro generaci pokusů
6  % H - H_X=H(X), funkce pro spočtení výkonnosti pokusů
7  % update - v=update(v,alpha,X,H_sort,index,N_b), funkce pro aktualizaci
     parametrů
8  % v_0 - cektor počátečních parametrů
9  % N_min - minimální počet pokusů v jedné iteraci CE metody
10 % N_max - maximální počet pokusů v jedné iteraci CE metody
11 % beta - počet přidaných pokusů při nesplnění podmínky růstu funkce H
12 % N_b - počet elitních snímků požitých pro aktualizaci parametrů rozdělení
13 % alpha - tlumící parametr při aktualizaci parametrů rozdělení
14 % c - počet iterací s maximální počtem pokusů pro ukončení metody
15 % d - počet iterací se stejnou maximální hodnotou pro ukončení metody
16 %%
17
18 v=v_0;
19 H_old=-inf*ones(1,d);
20 N_old=zeros(1,c);
21 gamma_old=-inf;
22 H_max_old=-inf;
23 vysledek=-inf;
24 N_sum=0;
25 for i=1:1000
26    N=N_min;
27    X=generate(v,N); % generování náh. pokusů
28    N_sum=N_sum+N;
29    H_X=H(X); % spočtení výkonnostní funkce
30    [H_sort,index]=sort(H_X); % seřazení od největšího po nejmenší
31    gamma=H_sort(N-N_b+1);
32    H_max=H_sort(end); % elitní snímek
33    if H_max>vysledek
34       vysledek=H_max; % dosavadní nejlepší řešení
35       argument=X(index(end),:); % a jeho argument
36    end
37    while ~((gamma>gamma_old)||(H_max>H_max_old)) % podmínka pro růst
        hodnot H_X
38       if same_values([H_max H_old]) % pokud v posledních d iteracích stejn
           é maximum
39          disp(['It:' num2str(i) ', celkem pokusů:' num2str(N_sum) '.
              Nalezeno spolehlivé maximum o hodnotě: ' num2str(vysledek)]);
40          num_iter=i;
41          return;
42       end
43       if N==N_max
44          if same_values([N N_old N_max]) % vyčerpaný počet iterací s max.
              počtem pokusů
45             disp(['It:' num2str(i) ', celkem pokusů:' num2str(N_sum) '.
                 Nalezeno nespolehlivé maximum o hodnotě: ' num2str(vysledek
                 )]);
```

```
46              num_iter=i;
47              return;
48           else
49              break;
50           end
51        end
52        N_temp=N;
53        N=min(N+beta,N_max);
54        N_temp=N-N_temp;
55        X=[X;generate(v,N_temp)]; %generace dalších pokusů
56        N_sum=N_sum+N_temp;
57        H_X=[H_X H(X(end-N_temp+1:end,:))]; %spočtení jejich výkonnosti
58        [H_sort,index]=sort(H_X);
59        gamma=H_sort(N-N_b+1);
60        H_max=H_sort(end); %nový elitní snímek
61        if H_max>vysledek
62           vysledek=H_max;
63           argument=X(index(end),:);
64        end
65     end
66     gamma_old=max(gamma,gamma_old);
67     H_max_old=max(H_max,H_max_old);
68     N_old=[N_old(2:end) N];
69     H_old=[H_old(2:end) H_max];
70     v=update(v,alpha,X,H_sort,index,N_b); % aktualizace parametrů
71  end
```

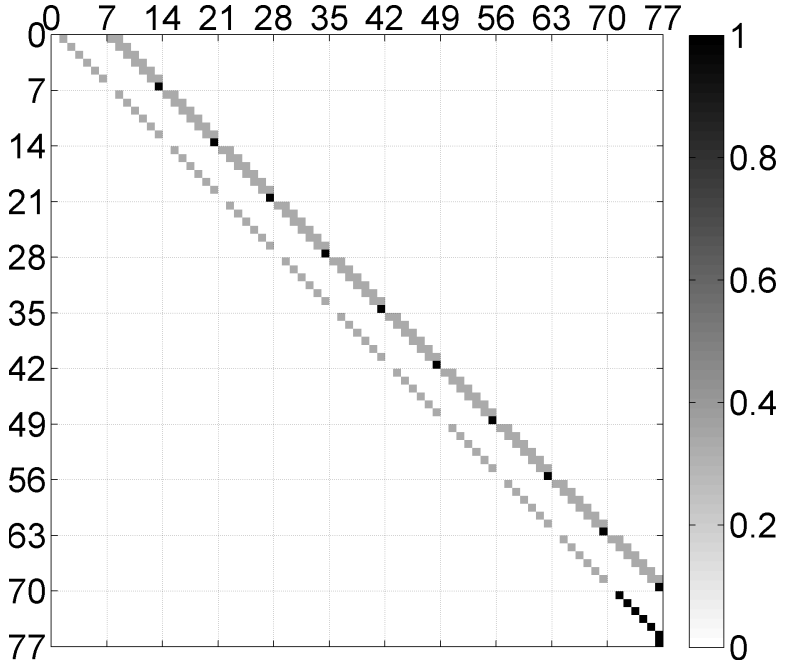
Výpis 2: FACE metoda pro optimalizace

D Problém umístnění n královen

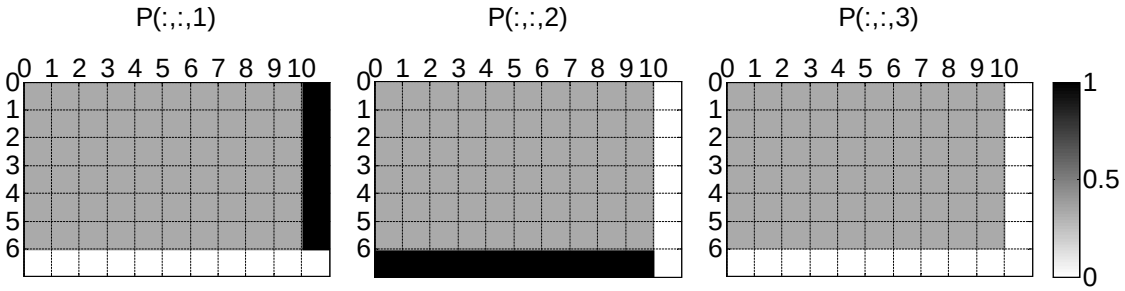


Obrázek D.1: Řešení umístnění královen pro $n = 16$

E Problém zarovnání sekvencí



Obrázek E.1: Matice přechodu pro MC pro úlohu na zarovnání sekvencí z příkladu 6.7



Obrázek E.2: Upravená matice přechodu z obrázku E.1

F Spojité optimalizace - benchmarkové úlohy

Řešíme úlohu na hledání globálního minima *Rosenbrockovy* a *trigonometrické* funkce (viz vzorce 7.1 a 7.2) na množině $\langle -2, 2\rangle^n$. Testování bylo provedeno s následujícím nastavením metod:

- CE: $N = 1000, N_b = 20, \alpha_\mu = 0.8, \alpha_\sigma = 0.9 - 0.9 \cdot (1 - 1/t)^6$,
- FACE: $N_{min} = 100, N_{max} = 10 \cdot N_{min}, N_b = 20, \alpha_\mu = 0.8, \alpha_\sigma = 0.9 - 0.9 \cdot (1 - 1/t)^6$,
- Matlab optimization toolbox: Algorithm: interior-point, MaxFunEvals: 10^7, MaxIter: 10^4, TolCon: 10^{-6}, TolFun: 10^{-6}, TolX: 10^{-10}, FinDiffRelStep: $1.49 \cdot 10^{-8}$.

Při použití CE a FACE metody volíme penalizační funkci

$$\phi(\boldsymbol{x}) = 100 \cdot \sqrt{\sum_{i=1}^{n} \min\{x_i + 2, 0\}^2 + \min\{2 - x_i, 0\}^2}$$

a jako počáteční parametry volíme $\widetilde{\boldsymbol{\mu}}_0 = (0, \ldots, 0)\,, \widetilde{\boldsymbol{\sigma}}_0 = (100, \ldots, 100)$.

Všechny testy byly provedeny 10 krát, v tabulkách lze vidět průměrné a nejhorší (maximální) výsledky.

F.1 Rosenbrockova funkce

n	$\overline{čas}$	$\overline{N_{sum}}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	0.10	249400	249.4	$2.94{\cdot}10^{-12}$	$7.83{\cdot}10^{-12}$	$2.33{\cdot}10^{-6}$	$5.70{\cdot}10^{-6}$
3	0.24	561200	561.2	$1.09{\cdot}10^{-10}$	$2.09{\cdot}10^{-10}$	$8.83{\cdot}10^{-6}$	$2.21{\cdot}10^{-5}$
4	0.47	990300	990.3	$1.37{\cdot}10^{-9}$	$2.20{\cdot}10^{-9}$	$3.66{\cdot}10^{-5}$	$8.18{\cdot}10^{-5}$
5	0.67	1249200	1249.2	$7.10{\cdot}10^{-9}$	$1.21{\cdot}10^{-8}$	$7.58{\cdot}10^{-5}$	$1.91{\cdot}10^{-4}$
6	1.14	2093200	2093.2	$1.29{\cdot}10^{-8}$	$1.98{\cdot}10^{-8}$	$7.68{\cdot}10^{-5}$	$1.95{\cdot}10^{-4}$
7	1.56	2690100	2690.1	$2.47{\cdot}10^{-8}$	$3.57{\cdot}10^{-8}$	$1.03{\cdot}10^{-4}$	$1.99{\cdot}10^{-4}$
8	2.07	3075200	3075.2	$7.49{\cdot}10^{-8}$	$1.49{\cdot}10^{-7}$	$2.76{\cdot}10^{-4}$	$6.71{\cdot}10^{-4}$
9	2.90	3986700	3986.7	$1.10{\cdot}10^{-7}$	$1.77{\cdot}10^{-7}$	$2.82{\cdot}10^{-4}$	$7.66{\cdot}10^{-4}$
10	3.97	5150900	5150.9	$1.37{\cdot}10^{-7}$	$2.75{\cdot}10^{-7}$	$3.39{\cdot}10^{-4}$	$6.51{\cdot}10^{-4}$

Tabulka F.1: Výsledky CE metody pro minimalizaci *Rosenbrockovy* funkce

n	$\overline{čas}$	$\overline{N_{sum}}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	0.67	275400	324.4	$2.68{\cdot}10^{-12}$	$5.11{\cdot}10^{-12}$	$2.23{\cdot}10^{-6}$	$5.02{\cdot}10^{-6}$
3	1.85	729660	843.6	$1.49{\cdot}10^{-12}$	$3.19{\cdot}10^{-10}$	$1.17{\cdot}10^{-5}$	$2.36{\cdot}10^{-5}$
4	1.88	708040	1272.8	$8.44{\cdot}10^{-7}$	$8.43{\cdot}10^{-6}$	$6.04{\cdot}10^{-4}$	$5.79{\cdot}10^{-3}$
5	2.05	772820	1533.3	$7.45{\cdot}10^{-9}$	$1.49{\cdot}10^{-8}$	$8.43{\cdot}10^{-5}$	$1.39{\cdot}10^{-4}$
6	2.60	955930	1731.9	$2.01{\cdot}10^{-8}$	$4.00{\cdot}10^{-8}$	$9.27{\cdot}10^{-5}$	$2.68{\cdot}10^{-4}$
7	3.25	1166960	1868.2	$4.04{\cdot}10^{-8}$	$7.89{\cdot}10^{-8}$	$1.48{\cdot}10^{-4}$	$3.45{\cdot}10^{-4}$
8	3.72	1304150	2060.2	$7.85{\cdot}10^{-7}$	$7.17{\cdot}10^{-6}$	$7.13{\cdot}10^{-4}$	$5.22{\cdot}10^{-3}$
9	3.90	1353690	2207.2	$1.85{\cdot}10^{-7}$	$6.88{\cdot}10^{-7}$	$4.98{\cdot}10^{-4}$	$1.37{\cdot}10^{-3}$
10	3.58	1233750	2382.2	$5.54{\cdot}10^{-6}$	$4.66{\cdot}10^{-5}$	$2.68{\cdot}10^{-3}$	$1.35{\cdot}10^{-2}$

Tabulka F.2: Výsledky FACE metody pro minimalizaci *Rosenbrockovy* funkce

n	$\overline{čas}$	$\overline{\text{poč.vyh. } H(x)}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	1.57	3654.7	15.8	$2.11{\cdot}10^{-11}$	$2.11{\cdot}10^{-11}$	$1.03{\cdot}10^{-5}$	$1.03{\cdot}10^{-5}$
3	1.73	4321.2	12.9	$3.71{\cdot}10^{-11}$	$3.71{\cdot}10^{-11}$	$1.25{\cdot}10^{-5}$	$1.25{\cdot}10^{-5}$
4	2.08	5337.1	13	$4.76{\cdot}10^{-11}$	$4.80{\cdot}10^{-11}$	$1.39{\cdot}10^{-5}$	$1.39{\cdot}10^{-5}$
5	2.12	6034	11.7	$5.34{\cdot}10^{-11}$	$5.34{\cdot}10^{-11}$	$1.46{\cdot}10^{-5}$	$1.46{\cdot}10^{-5}$
6	2.28	6589.2	10.4	$6.22{\cdot}10^{-11}$	$6.22{\cdot}10^{-11}$	$1.58{\cdot}10^{-5}$	$1.58{\cdot}10^{-5}$
7	2.42	6944.6	8.9	$6.42{\cdot}10^{-11}$	$6.42{\cdot}10^{-11}$	$1.60{\cdot}10^{-5}$	$1.60{\cdot}10^{-5}$
8	2.60	8004.3	9	$5.98{\cdot}10^{-11}$	$5.98{\cdot}10^{-11}$	$1.54{\cdot}10^{-5}$	$1.54{\cdot}10^{-5}$
9	3.13	9919.2	9.8	$6.60{\cdot}10^{-11}$	$6.60{\cdot}10^{-11}$	$1.62{\cdot}10^{-5}$	$1.62{\cdot}10^{-5}$
10	3.50	11198.5	9.8	$6.63{\cdot}10^{-11}$	$6.63{\cdot}10^{-11}$	$1.62{\cdot}10^{-5}$	$1.62{\cdot}10^{-5}$

Tabulka F.3: Výsledky Matlab opt. t. pro minimalizaci *Rosenbrockovy* funkce

F.2 Trigonometrická funkce

Testování bylo provedeno pro nastavení parametrů $\eta = 7, \xi = 1, \boldsymbol{x}^* = (1, \ldots, 1)$.

n	$\overline{čas}$	$\overline{N_{sum}}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	0.10	209500	209.5	1.0000	1.0000	$8.49{\cdot}10^{-7}$	$1.36{\cdot}10^{-6}$
3	0.15	299200	299.2	1.0000	1.0000	$4.18{\cdot}10^{-6}$	$5.70{\cdot}10^{-6}$
4	0.26	452500	452.5	1.0000	1.0000	$9.64{\cdot}10^{-6}$	$1.37{\cdot}10^{-5}$
5	0.43	681600	681.6	1.0000	1.0000	$1.82{\cdot}10^{-5}$	$2.35{\cdot}10^{-5}$
6	0.69	986000	986	1.0000	1.0000	$2.61{\cdot}10^{-5}$	$3.28{\cdot}10^{-5}$
7	1.18	1387100	1387.1	1.0000	1.0000	$3.76{\cdot}10^{-5}$	$4.56{\cdot}10^{-5}$
8	1.49	1822800	1822.8	1.0000	1.0000	$4.33{\cdot}10^{-5}$	$5.71{\cdot}10^{-5}$
9	2.16	2457200	2457.2	1.0000	1.0000	$5.54{\cdot}10^{-5}$	$6.76{\cdot}10^{-5}$
10	3.09	3255700	3255.7	1.0000	1.0000	$6.56{\cdot}10^{-5}$	$7.77{\cdot}10^{-5}$

Tabulka F.4: Výsledky CE metody pro minimalizaci *trigonometrické* funkce

n	$\overline{čas}$	$\overline{N_{sum}}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	0.61	263760	327.5	1.0000	1.0000	$5.19{\cdot}10^{-7}$	$1.17{\cdot}10^{-6}$
3	1.14	442010	524.2	1.0000	1.0000	$3.12{\cdot}10^{-6}$	$4.35{\cdot}10^{-6}$
4	1.82	668640	776.4	1.0000	1.0000	$9.91{\cdot}10^{-6}$	$1.44{\cdot}10^{-5}$
5	2.39	850900	972.2	1.0000	1.0000	$2.20{\cdot}10^{-5}$	$2.88{\cdot}10^{-5}$
6	2.99	1015290	1153.1	1.0000	1.0000	$2.94{\cdot}10^{-5}$	$3.79{\cdot}10^{-5}$
7	3.55	1172110	1324.5	1.0000	1.0000	$4.51{\cdot}10^{-5}$	$5.79{\cdot}10^{-5}$
8	4.12	1313030	1477.3	1.0000	1.0000	$5.41{\cdot}10^{-5}$	$6.81{\cdot}10^{-5}$
9	4.77	1482130	1657.6	1.0000	1.0000	$6.78{\cdot}10^{-5}$	$8.18{\cdot}10^{-5}$
10	5.32	1588990	1773.7	1.0000	1.0000	$8.45{\cdot}10^{-5}$	$1.07{\cdot}10^{-4}$

Tabulka F.5: Výsledky FACE metody pro minimalizaci *trigonometrické* funkce

n	$\overline{čas}$	$\overline{\text{poč.vyh. } H(x)}$	$\overline{Iter}$	$\overline{H^*}$	$\max H^*$	$\overline{\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|}$	$\max \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|$
2	1.03	2549.9	8.3	1.0897	1.8974	0.0947	0.9472
3	0.91	2765.8	6.5	1.1795	1.8974	0.1894	0.9472
4	1.36	3683.5	10.1	1.6731	4.5898	0.5406	1.8945
5	1.72	4156.7	9.7	1.3141	3.2434	0.2837	1.4976
6	1.95	5117.1	10.6	1.5833	3.2435	0.4667	1.4977
7	3.22	8777.6	17.3	1.7179	4.1410	0.5556	1.7721
8	3.87	10113.6	17.8	1.8076	4.1408	0.6101	1.7720
9	5.19	16150.6	25.2	2.8397	4.5898	1.1835	1.8945
10	3.69	11743	15.3	3.6474	7.2819	1.4194	2.5061

Tabulka F.6: Výsledky Matlab opt. t. pro minimalizaci *trigonometrické* funkce

G Příloha na CD

Přiložené CD obsahuje:

- elektronickou kopii diplomové práce
- veškeré zdrojové kódy vzniklé jako součást této práce
- originály vložených obrázků